Universitat de Lleida

TREBALL FINAL DE GRAU

Estudiant: Roger Capdevila Castells

Titulació: Doble Grau en Enginyeria Informàtica i ADE

Títol de Treball Final de Grau: **Banking applications**

Director/a: **Santi Martínez Rodríguez**

Presentació

Mes: Juliol

Any: 2019

Index

Table of figures

I would like to thank Santiago Martinez and Carles Torras, who are the directors of this project. I would like to express my sincere gratitude to Dolors Navarro and Carles Torras too, because all hours they have spent helping to develop the database, without their help this project wouldn’t have been possible to do. Also I would like to thank Aimee Plunkett for the effort done reading all project once it was done in order to help about usage of English, and my family for all support I have received from them.

Abstract

In a globalized world, where distances don't mean so much, purchases can be done from one side of the world to the other. In a world where we can have immaterial assets whose prices depend on how many people want them, nowadays work is not the only thing people use to do to earn money. Years ago only people with a lot of money and companies could afford investments, however nowadays we can have a 10€ investment. Also our saving capacity has increased and even the working class can invest in many different assets, sold and stored in many different financial institutions, and everyone want to have a control of their assets, their actual values and profits.

All this investments, assets, institutions and markets are producing an enormous amount of information, also needed to have this control that people are looking for about their investments, and this is what gives sense to this project. What is going to be explained is a way to store and sort all information that users produce, buying and selling financial instruments, in order to build an application with a friendly interface and computing capacity, to help the user to take decisions.

Key words: financial instrument, account, transaction, database, foreign key, trigger.

1. Introduction

With the Internet our needs and capacities have been increased exponentially. In the same way we can speak with people around the world at the same time, we can visit websites, see videos and do some purchases just from our desk.

All these talks, visits, videos and purchases mean information. By 2025, it's estimated that 463 exabytes of data will be created each day which means 462,000 million gigabytes per day. Most of this information is stored in the cloud which means they are in company's hard disks connected to the net.

But what happens when we want to store some data, in a specific way to build an application which is going to use that data? And what happens with the security?

One of the most important things we care about security is our financial information, which creates a lot of information with complex relationships.

So in this project we are going to design a database to store all information about financial assets, transactions made with them and more information related as markets and banking entities. The result will be a SQL which will build the database designed when it is executed.

1.2 Needs detected

Nowadays, the economy and finances have reached the family and personal environment. This has meant that, most people can manage their capital and assets, depositing them in many banks and financial institutions. This fact forces us, to be able to quantify all what we have, to consolidate data from many different institutions.

We should say that, with online banks, this work has become easier to do, but every single bank gives us many different products, with different risks, financial options and rates. It means that, even though we can manage all our resources from home, with a computer and an internet connection, we still don't have a tool that allows us to have a global vision of all those resources, to make it easier to make a decision about what we should do with our resources.

We usually find ourselves in front of large spreadsheets where there are multiple rows of collected data, accumulated through time, where it is hard to analyse, interpret or find an exact transaction or yield, because we have more than one account, in more than one bank, where the data is organized in different ways. This makes it impossible to have a global vision of all what we have, the actual state of every asset.

In addition, if we think about the people and companies who are managing all assets from other people and companies, those spreadsheets make that work too hard and difficult to make it the best they could.

Usually, these people and companies think about a software to make their job easy, like an ERP, but these software products are not meant to do that job.

Therefore, the need we've detected, is the need to have a tool which could help us manage many accounts, from many banks and financial entities and make decisions about all our assets and money.

1.3 Goals

After we have detected the need of a tool to manage personal capital and assets, or from a company, our goal is to make a software which allows us to do that work.

However, the whole software is too big for a single project, so we have divided it in two parts and we'll develop the first one:

1. Design of a database where storing all data to make it easy to build an application which will need all that data sorted in the best possible way.
2. Build an application that can give a global vision of many assets from many accounts in many banks.

1.4 Structure and development approach

In order to give an answer, to the need described previously, and develop a software which allows us to have a global vision of our financial and real estate assets, and be able to make decisions about our own familiar economy and also elaborate a business plan to make an economic benefit of that product, it is needed to know how it is going to be structured, all information it will have to stored, the steps that are needed and their order.

As it has been said in the previous section, we are going to develop and implement a database to store all the data needed to manage personal, and enterprise, financial and real estate assets.

This work is divided into eight parts, which are listed below:

1. Detect and define all information needed to be stored, all its attributes, and its different types as well.
2. Based on the knowledge of all the information needed to store, design a database model which allows keeping the consistency of a well-designed database, and make sure no information is mixed or gets into conflict between them.

   This design has to make it easy to maintain all of the information in the database, and ensures all queries are not too expensive, in terms of time and needed resources.

   In case some information is repeated in the database, it must be because it's completely needed to improve the efficiency of some queries.

3. Determine all triggers needed to ensure data stored in all tables is updated after a transaction, or an action made in a table.
4. Create a schema of the database, with its tables and all relations between them.
5. Create the database.
6. Create a web application that interacts with the database, to make it easy for anyone to test all actions that must be able to do so on the database.
7. Test every single table in the database. For every table, we must be able to read the data stored in it, and also insert, delete and modify any record.

2. Information needed to store

Computer science tries to help us make difficult and big calculations, to make predictions, and also build artificial intelligences, but for it we first need to understand what are we going model. This is critically important for creating accurate results of our software.

This is called requirements engineering, and it is the first part of any computer science project.

In our project we'll build a database about financial instruments, transactions and accounts, so in this section will define all of them and their attributes.

2.1 Financial instruments

In accordance with the definition, by the International Accounting Standards (IAS), "financial instruments are any contract that generates a financial asset in a company, and a financial lability or equity instrument in another. In our case, it is each one of the different financial products types, which we will store in the database." [1]

In this section, we will define every single financial instrument which will be able to be stored in our database. We will also enumerate their attributes. Since instruments share many attributes, these will be defined in the next section.

Financial instruments are divided in two big types, depending on what is their price based on.

2.1.1 Cash Instruments

"Cash instruments are financial instruments whose value is ascertained directly by markets. Cash instruments can be classified into two types as securities and other cash instruments such as loans and deposits. Securities are readily transferable, whereas loans and deposits can be transferred only when both borrower and lender agrees for the transfer. Cash instruments often offer complete capital security, but subject to credit risk. As a result, the capital value of cash instruments will not fluctuate if interest rates fluctuate. Since cash instruments are highly liquid, it can be used by institutions with very long-term liabilities, to meet their immediate cash flow needs." [2]

Currency

"A currency is the money which is used in a particular country at a particular time." [3]

The term currency refers to the metal or paper money used to obtain goods, products and services. It is considered a financial instrument since, its value, it suffers variations for each commercial and financial movement from a country to another, and therefore it is bought and sold to obtain economic returns taking advantage of these variations. This instrument has two purposes:

- Make payments, materialize income and value other assets, financial or otherwise.
- Get economic profitability by buying, accumulating and subsequently selling in other economic areas.

Currency attributes are:

- Code
- Name
- Value
- Issuing entity
- Product type
- Instrument class
- Sector
- Quoted
- Issuing date
- Counterpart

Fund

"An investment fund is a supply of capital belonging to numerous investors used to collectively purchase securities while each investor retains ownership and control of his own shares. An investment fund provides a broader selection of investment opportunities, greater management expertise and lower investment fees than investors might be able to obtain on their own.

With investment funds, individual investors do not make decisions about how a fund's assets should be invested. They simply choose a fund based on its goals, risk, fees and

other factors. A fund manager oversees the fund and decides which securities it should hold, in what quantities and when the securities should be bought and sold." [4]

Elements that intervene in a found:

- Management Company: they are, often, anonymous and their exclusive corporate purpose is the administration, management and representation of collective investment institutions, in this case of investment fund.
- Depositary entity: it is an authorized institution in accordance with the securities Market Lay and registered in the CNMV to carry out this activity its main functions are to safeguard the values that make up the equity of found, to control the correct operation of managing entity and to authorize the payments to the participants.
- The value of participation: the assets are divided daily by the number of shares in circulation, determining the participation price that will be applied to purchases/sales of the day.

Funds attributes are:

- Code
- Name
- Value
- Issuing entity
- Sector
- Instrument class
- Product type
- Quoted
- Quote market
- Currency
- Issuing date
- Managing entity, or depositary entity
- Composition

Fixed-income security

"Fixed-Income securities are debt instruments that pay a fixed amount of interest—in the form of coupon payments—to investors. The interest payments are typically made semi-annually while the principal invested returns to the investor at maturity. Bonds are the most common form of fixed-income securities. Companies raise capital by issuing fixed-income products to investors." [5]

Among the assets of this type of investment are:

- Bond: It is a financial instrument of debt used by private and government entities. These can also be used by supranational entities (ECB, CPA, etc.) and aim to obtain funds directly from the financial markets. They are titles usually in the name of the carrier that are usually traded on a stock market or stock exchange. The issuer agrees to return the principal capital together with the interests.
- Obligation: It is a financial instrument that represents a part of the debt of a company, or state. These securities give the buyer an economic rich consisting of a coupon income and the payment of the money at the end of the stipulated term. The obligation works as a bound, but the difference is that the bound usually has a shorter term (up to 5 years) and the obligation has a longer term (more than 5 years).
- Promissory note: It is an accounting document that contains the unconditional promise of a person (debtor), that he will pay to a second person (beneficiary or creditor) a certain amount of money in a certain period of time.
- Bill of exchange: It is a credit of formal value and complete, that contains an unconditional, and abstract order to make to pay at maturity to the policyholder, a sum of money, in a certain place.
- ETN: Acronym of exchange-traded note. It is a financial asset that places the price of an action, a raw material or index, in a limited period of time, usually in the very long term, and at maturity it is paid to investor the revaluation of the stock, raw material or index.

Fixed-income security attributes are:

- Code
- Name
- Value

- Issuing entity
- Sector
- Instrument class
- Product type
- Quoted
- Quote market
- Currency
- Issuing date
- Accrual date
- First coupon date
- Expiration date
- Nominal
- Interest rate
- Payday
- Days of calculation
- Nominal reduction
- Counterpart
- Physic Title
- First call
- PUT

Property

It can be said that there are two types of properties, depending on their purpose.

A property can just be the house which someone buys to live in it, or an inherited land, whose purpose is not to have an economic return.

However, a property can be bought as an investment.

“Investment property is real estate property that has been purchased with the intention of earning a return on the investment, either through rental income, the future resale of the property or both. An investment property can be a long-term endeavour or an intended short-term investment such as in the case of flipping, where real estate is bought, remodelled or renovated, and sold at a profit.” [7]

Property attributes are:

- Code
- Name
- Value
- Issuing entity
- Sector
- Instrument class
- Product type
- Currency
- Issuing date
- Land register value

Variable-income security

"The term variable-income security refers to investments that provide their owners with a rate of return that is dynamic and determined by market forces. Variable-income securities provide investors with both greater risks as well as rewards. Also known as variable-rate securities, variable-income securities are typically valued by investors looking for higher returns than those offered by fixed-income securities. The classic example of a variable-income security is common stock, which can offer investors virtually unlimited up-side growth as well as the complete loss of principal. In exchange for this risk, investors in these securities demand higher returns than their fixed-income counterparts." [6]

Among the assets of this type of investment are:

- Share: A property title by a part of the capital of a company. The type of companies that have their capital represented by shares are so-called Public Limited Corporations (PLC). The possession of a share gives political rights (preferential right to participate in capital increases, and right to vote at the general meetings, when the number of shares, in ownership, corresponds to the minimum amount stipulated in the bylaws), and economic ones. The shares can be sold, whenever its owner desires.

- PSR: Acronym of Preferential Subscription Rights. It is a right that is obtained by each share owned by a company, when it makes a capital increase. These rights are issued to not harm the current shareholders and give them the opportunity to maintain the same percentage of participation in the capital of the organization. These rights have a price, are listed on the stock market, and can be used or sold.
- Warrant: This is a corporate title which offers the right, but not the obligation, to buy or sell shares of a company at a fixed price for a given period. The fact of carrying out the transaction is called "exercising" the warrant. Each warrant specifies the number of shares that the tenant can buy, the exercise price and the due date.

  Warrants are also traded on the stock market and can be bought and sold. If a warrant is for purchasing, it receives the name of "call warrant", if it is for selling, it is called "put warrant".
- ETF: Acronym for Exchange-Traded Funds. They are investment funds that are listed on the stock market, as well as shares, and can be bought and sold during a session at the existing price at any time, without waiting for closure of the market to know the liquid value at which the subscription/reimbursement is made.

  ETFs are characterized because the main objective of their investment policy is to reproduce a certain stock or fixed income index, and their shares are admitted to trading on the stock market. The difference between these and the "index fund" is that they allow acquisition or sale of the participation, whatever it is desired, throughout the daily negotiation period.

Variable-income security attributes are:

- Code
- Name
- Value
- Issuing entity
- Sector
- Instrument class
- Product type
- Quoted
- Quote market

- Currency
- Issuing date
- Nominal
- Physic title

2.1.2 Derivative instruments

"Derivative instruments are financial instruments which gain their value or change in value from the value and features of the underlying assets. They are treated as an alternative form of investment. They can be classified into exchange-traded derivatives and over-the-counter (OTC) derivatives. They are excellent means for maximizing return on an investment, and for successfully hedging a financial portfolio. Usually, derivatives instruments are categorized on the basis of the relationship between the underlying asset and the derivative; the type of underlying asset; the market in which they trade; and their pay-off profile. Derivative instruments are also associated with risk because any event or market movement that has an adverse impact on the value of the underlying security will also cause the instrument to lose value." [8]

Derivative products can be, for example:

- Options: It is a financial derivative instrument that is stablished in a contract which gives to its buyer, the right, but not the obligation, to buy or sell an underlying asset, at a predetermined price (strike) until a specific date (maturity). There are two types of options, purchase option (call) and sell option (put).
- Future contract: It is a contract that obliges the parts to buy or sell a certain number of goods, and underlying assets at a specified future date, at a price stablished in it. These contracts are negotiated in, the so-called, "futures market".

Derivative assets attributes are:

- Code
- Name
- Value
- Issuing entity
- Sector
- Instrument class

- Product type
- Quoted
- Quote market
- Currency
- Issuing date
- Expiration date
- Nominal

2.2 Financial instrument's attributes

An attribute is a quality or a part of something which makes it different, sometimes unique.

Every financial instrument has many attributes, some of them are shared with all other instruments, some are shared only with some other instruments, and there are some attributes which we only can find in one instrument.

In this part will be defined all attributes we have seen in each financial instrument in the last section.

Accrual date

"Accruals are earned revenues and incurred expenses that have an overall impact on an income statement. They also affect the balance sheet, which represents liabilities and non-cash-based assets used in accrual-based accounting. These accounts include, among many others, accounts payable, accounts receivable, goodwill, future tax liabilities and future interest expenses." [9]

Then the accrual date is the date when revenues are received, and expenses are paid.

Code

We can see this attribute in every asset because this is the attribute which will identify every single asset. We can say that this attribute is the main identifier of every asset.

Every code is unique, and it is impossible to find two assets with the same code, even these assets are different instrument types.

The code can be a string, or an integer, and it never will be a Boolean or date.

However, in the database we will build, this attribute can be called in different ways like product code, ISIN or ISIN related.

Composition

As we have seen before, a fund is an asset which its individual investors do not make decisions about how assets should be invested. They simply choose a fund based on its

goals, risk, fees and other factors. A fund manager oversees the fund and decides which securities it should hold, in what quantities and when the securities should be bought and sold.

So, the fund composition, is the list of securities that a fund holds, and the percentage that represents each one in the fund.

Counterpart

"A counterparty is the other party that participates in a financial transaction, and every transaction must have a counterparty in order for the transaction to go through. More specifically, every buyer of an asset must be paired up with a seller who is willing to sell and vice versa. […]

The term counterparty can refer to any entity on the other side of a financial transaction. This can include deals between individuals, businesses, governments, or any other organization. Additionally, both parties do not have to be on equal standing in regards to the type of entities involved. This means an individual can be a counterparty to a business and vice versa. In any instances where a general contract is met or an exchange agreement takes place, one party would be considered the counterparty, or the parties are counterparties to each other.

A counterparty introduces counterparty risk into the equation. This is the risk that the counterparty will be unable to fulfil their end of the transaction. However, in many financial transactions, the counterparty is unknown and the counterparty risk is mitigated through the use of clearing firms. In fact, with typical exchange trading, we do not ever know who our counterparty is on any trade, and often times there will be several counterparties each making up a piece of the trade." [10]

Currency

All financial assets have an economic value. This value is the amount of money which a seller will receive if the asset is sold. But this amount of money has to be expressed in a concrete currency, so this attribute will identify the currency in which the asset is valuated.

Expiration date

Also called maturity date, “is the date on which the principal amount of a note, draft, acceptance bond or another debt instrument becomes due and is repaid to the investor and interest payments stop. It is also the termination or due date on which an instalment loan must be paid in full.

The maturity date defines the lifespan of a security, informing you when you will get your principal back and for how long you will receive interest payments. However, it is important to note that some debt instruments, such as fixed-income securities, are "callable," which means that the issuer of the debt is able to pay back the principal at any time. Thus, investors should inquire, before buying any fixed-income securities, whether the bond is callable or not.” [11]

In derivatives, “An expiration date […] is the last day that a derivative, such as options or futures, is valid. On or before this day, investors will have already decided what to do with their expiring position.

Before an option expires, its owners can choose to exercise the option, close the position to realize their profit or loss, or let the contract expire worthless.” [12]

First call

“The earliest date on which a security may be redeemed by the issuer. This date is partic ularly important to an investorwho holds a security that is selling above or near its call p rice. The first call date is likely to be either five years or tenyears after the date of issue; however, the timing varies by bond issue. Information regarding the first call date may befound on the bond certificate or it may be obtained from the brokerage firm holding t he security. Bonds selling at apremium are often quoted at the yield to first call.” [13]

First coupon date

This is a fixed income assets attribute which determines the date of the first coupon payment.

Instrument class

"An asset class is a grouping of investments that exhibit similar characteristics and are subject to the same laws and regulations. Historically, the three main asset classes have been equities (stocks), fixed income (bonds) and cash equivalent or money market instruments. Currently, most investment professionals include real estate, commodities, futures, other financial derivatives and even cryptocurrencies to the asset class mix. […]

Asset classes and asset class categories are often mixed together. There is usually very little correlation, and in some cases a negative correlation, between different asset classes. This characteristic is integral to the field of investing.

Financial advisors view investment vehicles as asset class categories that are used for diversification purposes. Each asset class is expected to reflect different risk and return investment characteristics and perform differently in any given market environment. Investors interested in maximizing return often do so by reducing portfolio risk through asset class diversification.

Financial advisors focus on asset class as a way to help investors diversify their portfolio. Different asset classes have different cash flows streams and varying degrees of risk. Investing in several different asset classes ensures a certain amount of diversity in investment selections. Diversification reduces risk and increases your probability of making a return." [14]

Issuing date

Issuing date, or date of issue is "used in the context of stocks to refer to the date trading begins on a new stock issued to the public." [15]

Issuing entity

The issuing entity is the entity which put a financial asset in the market.

"The role of an issuing entity is to provide securities for investors to purchase, usually as a means of generating a profit for itself or its owners. Securities are tradable financial instruments with value and include broad categories of investments, such as stocks and bonds. In each case, the issuing entity sells a financial instrument to investors through a

market, later paying interest or allowing investors to sell their securities to one another freely." [16]

Interest rate

"The interest rate is the amount a lender charges for the use of assets expressed as a percentage of the principal. The interest rate is typically noted on an annual basis known as the annual percentage rate (APR). The assets borrowed could include cash, consumer goods, or large assets such as a vehicle or building." [17]

Land register value

This is the value of a property, or a real estate, in the land register.

It doesn't mean that when the property will be sold, it will be sold for that value, even if it didn't use to be lower than the land register value. It can be depending on many different factors like the zone where the property is, how many people want to buy the property or how quick the actual owner want to sell it.

Managing entity

This is a fund attribute. It is the entity who manages a fund, even if it has no participation in this fund.

"Funds management is the overseeing and handling of a financial institution's cash flow. The fund manager ensures that the maturity schedules of the deposits coincide with the demand for loans. To do this, the manager looks at both the liabilities and the assets that influence the bank's ability to issue credit." [18]

Name

We can also see this attribute in every financial instrument. It is an identifier, but it can be repeated, even then it will be useless, because it is not the main identifier as the code.

For example, when we talk about a currency, its name makes us easy to identify which currency we are talking about.

Nominal

Also called principal, "[…] is a term that has several financial meanings. The most commonly used refers to the original sum of money borrowed in a loan or put into an investment. Similar to the former, it can also refer to the face value of a bond. […]

When you make monthly payments on a loan, the amount of your payment goes first to cover accrued interest charges, and the remainder is applied to your principal. Paying down the principal of a loan is the only way to reduce the amount of interest that accrues each month." [19]

Nominal reduction

Also called principal reduction, nominal reduction "[…] is a decrease granted toward the principal owed on a loan, typically a mortgage. A principal reduction can be obtained to decrease the outstanding principal balance on a loan and provide relief for a borrower. Principal reduction is normally deployed to prevent foreclosures on properties, which may be more costly to financial institutions than a reduced principal owed to them." [20]

Payday

Through time there is a series of payments, like when a loan is paid back, the payday is the day when the payment will be done, for example, the first of the month.

Physic title

This is a Boolean attribute of fixed and variable income assets which indicates whether there exists a physical document which represents the asset owned.

Product type

This attribute will have a table for itself. This is because the type can have a lot of information.

As a product type we will save a type ID, just to identify every type, the user will have to create, and three string fields, where the user will be able to store all information of this type that she/he would like to store, for example if it's a saving or investment product, its risk, or any considerations that the user would like to store about the instrument.

PUT date

"A PUT is an options contract that gives the owner the right, but not the obligation, to sell a certain amount of the underlying asset, at a set price within a specific time. The buyer of a put option believes that the underlying stock will drop below the exercise price before the expiration date. The exercise price is the price that the underlying asset must reach for the put option contract to hold value." [21]

So, the PUT date is the specific date when the option can be exercised.

Quoted

This will be a Boolean attribute, which when its value is true, will mean that we can find the asset in a quote market.

Quote market

"A financial market is a broad term describing any marketplace where trading of securities including equities, bonds, currencies, and derivatives occur. Some financial markets are small with little activity, while some financial markets […] trade trillions of dollars of securities daily.

Financial market prices may not indicate the true intrinsic value of a stock due to macroeconomic forces like taxes. In addition, the prices of securities are heavily reliant on informational transparency to ensure efficient and appropriate prices are set by the market.

The stock market is a financial market that enables investors to buy and sell shares of publicly traded companies. The primary stock market is where new issues of stocks are first offered. Any subsequent trading of stock securities occurs in the secondary market." [22]

Sector

An economic sector is “a division of a country's population based upon the economic area in which that population is employed. Many economists recognize the following five economic sectors; the primary sector which includes agriculture, mining and other natural resource industries; the secondary sector covering manufacturing, engineering and construction; a tertiary sector for the service industries, the quaternary sector for intellectual activities involving education and research and the quinary sector reserved for high level decision makers in government and industry.” [23]

Value

The value of an asset is the amount of money that will be received if the asset is sold.

That value is always measured in a currency, which also can be valued in other currencies.

2.3 Accounts where financial assets will be stored

When someone buys any financial asset, these assets have to be reflected in an account, with the amount held and their value. Holders can see all their financial assets in their portfolio from looking at their financial accounts

This section will define every different account we have, to store all financial assets which are defined before, and enumerated their attributes which will be defined in the next section.

We have to notice that all accounts that will be defined, will not store all transactions belonging to them. Later, will be explained why and how all transactions will be registered in the database.

Current account

“The current account records the payments for goods and services, plus investment income and transfers, between an economy and the rest of the world. Payments coming into an economy are called credits and payments leaving an economy are called debits.

The current account is considered the most significant account for an economy, although it does not include investment flows, and hence does not measure all financial flows between countries - which are included in the overall 'balance of payments'.” [24]

This account is maybe the more useful in the database, in terms of data stored in that table, since in every current account, it is only permitted to store a single currency. Anyway, any purchase and sell with financial assets needs to be linked with a current account, so the money used to make these transactions, must be stored in a current account.

Current account attributes are:

- IBAN
- Currency
- Sing up date
- Unsubscribe date
- Banking entity

- Balance

Financial assets account

This account will be used to store data about the derivatives we own.

This account will only have a list of all those derivative assets we still own. It means that if we have sold an asset it will not be in this list.

Financial assets account attributes are:

- Numbering
- ISIN
- Currency
- Banking entity
- Sing up date
- Unsubscribe date
- Balance

Participant account

This account will be used to store data about fund assets we own.

This account will only have a list of all assets we still own. It means that if we have sold an asset it will not be in this list.

Participant account attributes are:

- Depositary and numbering
- Banking entity
- Sing up date
- Unsubscribe date
- Balance

Values account

This account will be used to store data about the fixed and variable income assets we own.

This account will only have a list of all those assets we still own. It means this list will not have all assets we have owned in the past, but we already don't own.

Values account attributes are:

- ISIN
- CCV
- Banking entity
- Sing up date
- Unsubscribe date
- Balance

Reference register

This account will be used to store data about the properties we own.

This account will only have a list of all those properties we still own. It means this list will not have all properties we have owned in the past, but we already don't own.

Reference register attributes are:

- Land registry ID
- Currency
- Sing up date
- Unsubscribe date

2.4 Accounts attributes

As in financial attributes, accounts have many attributes, some of them are shared with all other accounts, some are shared only with some other accounts, and there are some attributes which we can only find in one account.

In this part will be defined all attributes we have seen in each account, in the last section.

IBAN

"An international bank account number (IBAN) is a standard international numbering system, developed to identify bank accounts from around the world. Banks in Europe originally developed the system to simplify transactions involving bank accounts from other countries.

An IBAN does not replace a bank's own account number; rather, this is an additional number, with further information, which helps in identification for overseas payments." [25]

How a IBAN looks like:

[26]

Currency

This attribute indicates in which currency is valuated the assets that there are in an account.

Balance

"An account balance is the amount of money in a financial repository, such as a savings or checking account, at any given moment. It can also refer to the total amount of money owed to a third party, such as a credit card company, utility company, mortgage banker or other type of lender or creditor. The account balance is always the net amount after factoring in all debits and credits. Debts can sometimes be considered negative account balances; for example, when there is an overdraft on a checking account." [27]

Also, in our database, the balance will indicate the amount of a financial asset we have, in financial assets account, participant account and values account.

Sign up date

The sign up date is the date that an account is opened by a person, a company or an organization.

Unsubscribe date

The unsubscribe date is the date when an account is closed, cancelled or removed by its owner.

Numbering

This attribute can be found only in financial assets accounts, where the main identification of these type of accounts are.

ISIN

Acronym of International Securities Identification Number.

"ISINs are used to identify most types of financial instruments, include equity, debt and derivatives.

The ISIN code consists of a total of 12 characters as follows:

The first two characters are taken up by the alpha-2 country code as issued in accordance with the international standard ISO 3166 of the country where the issuer of securities, other than debt securities, is legally registered or in which it has legal domicile. For debt securities, the relevant country is the one of the ISIN-allocating National Numbering Agency. In the case of depository receipts, such as ADRs, the country code is that of the organization that issued the receipt, rather than instead of the one that issued the underlying security.

The next nine characters are taken up by the local numbering code of the security concerned. Where the national number consists of fewer than nine characters, zeros are inserted in front of the number so that the full nine spaces are used.

The final character is a check digit computed according to the modulus 10 “Double-Add-Double” formula.

For ISINs of OTC derivatives, which will be allocated by a single global numbering agency, the initial two digits will use a custom “EZ” code. The ISIN will be generated on a de novo basis, with no reference to previous or other codes.” [28]

So, the reason to have it as an attribute in some accounts, is because the instrument types that these accounts can allocate, need an account for every asset bought. Then this ISIN will be related to ISIN in transactions done in the account that will refer to assets codes.

Banking entity

The banking entity attribute indicates in which bank, or financial institution, an account is opened.

Depositary and numbering

This attribute is used in participant accounts as the main identification number.

CCV

This attribute is used in a values accounts as the main identification number.

Land registry ID

"The land registry ID is the official, and mandatory, identification of real state. It consist of an alphanumeric code […] so that every property must have a single reference that allows it to be placed unequivocally […]." [29]

Therefor this ID is used to identify every single real state we own.

2.5 Transactions

In commerce "exchange of goods or services between a buyer and a seller. Every transaction has three components: (1) transfer of good/service and money, (2) transfer of title which may or may not be accompanied by a transfer of possession, and (3) transfer of exchange rights." [30]

Also, in the banking sector, a transaction is an "activity affecting a bank account and performed by the account holder or at his or her request." [30]

In our database we must store every single transaction that a user has done with his/her assets, so this section will be enumerate and define all the information we must store about every type of transaction, for each instrument we will be able to store in the database.

Currency transactions

The first data we need to store about transactions, in order to ensure that we can make two transactions at the same time and store both of them separately, is an identifier for each transaction. This identifier will be an integer and it will be mandatory to give an ID in every single transaction.

In real life, all currencies will be stored in at least, one current account. This means that every non-empty account will have, at least, a transaction related with it. So, the second data we need to store about these type of transactions, is the IBAN of the transaction related account.

We can also make transactions between two different current accounts. This is the reason why it will be possible to store a second IBAN, even this second IBAN will not be mandatory to fill.

A transaction will be done with a currency, for example buy something, pay a bill or transfer money from an account to another, and the currency with which we are making this transaction does not need to be done with the same currency that the current account is saving our money. Then we will store the currency which we are making the transaction.

Also, we can buy a currency, so we will need to be able to store both currencies, even the second one will not be mandatory to fill.

The most important thing in a currency transaction is the amount of the transaction and the database will store it as well.

Sometimes the banking entity where the current account is allocated charges some money to make the transaction, which makes the transaction more expensive. With the aim of separating what is the amount of the transaction and its cost, a transaction will also store the bank commission.

Another important attribute of a transaction is the date when it is done, and we will call it the transaction date.

With currency transactions we can make a transaction with cash, like saving money to a current account going to the bank and giving them the cash, or we can pay a purchase with a debit card. In the same way we can pay a purchase with a credit card, so the database will store it, if a transaction is a cash transaction or it isn't. To do so, it will be a Boolean mandatory field which when its value is true, it will mean it is a cash transaction, and if its value is false it will mean it is not a cash transaction.

In order to be able, in the future, to do a proper analysis of our currency transactions, it will be so useful to know if a transaction is made because of a purchase, a bill, rent, mortgage or loan payment, or a salary payment. For this reason, it will be mandatory to store the transaction type, which will be a string that will link a currency transaction and another entity that we will call "Transaction types". This entity will have the accountant information we need to store about each currency transaction.

One more attribute of the currency transactions is the value date, even it will not be a mandatory field to fill, the database will allow us to save its value if it is necessary.

"A value date is a future date used in determining the value of a product that fluctuates in price. Typically, you will see the use of value dates in determining the payment of financial products and accounts where there is a possibility for discrepancies due to differences in the timing of valuation. […]

When a payee presents a check to the bank, the bank credits the payee's account. However, it could take days until the bank receives the funds from the payor's bank, assuming the payor and payee have accounts with different financial institutions. If the payee has access to the funds immediately, the receiving bank runs the risk of recording a negative cash flow. To avoid this risk, the bank will estimate the day it will receive the money from the paying institution, and hold the funds in the payee's account until the expected day of receipt. In effect, the bank will post the amount of the deposit for a couple of days, after which the payee can use the funds. The date the funds are released is referred to as the value date.

Likewise, when a wire transfer, is made from an account in one bank to an account in another bank, the value date is the date on which the incoming wire becomes available to the receiving bank and its customer." [31]

We also will save in every transaction its concept and origin, which is the transaction code, if the origin of the currency transaction is a financial assets transaction.

Even currencies are used to set a price to assets, food, cars, or anything we can buy, they also have a price based on other currency. So, when we will be making a currencies purchase or sell, we will be able to set the price of that currency we are buying or selling, based on our reference currency. If the transaction is not a currencies purchase or sell, the price of the currency will always be 1.

In the future we might buy forward start options, and they will be paid with currencies, so we will be making a currency transaction too. A forward start option "A forward start option is an exotic option that is purchased now but becomes active later with a strike price determined at that time. The maturity date and underlying asset are also fixed at the time of purchase. […]

At initiation, a forward start option contract spells out all the defined characteristics relevant to any option, except for its strike price. The maturity date, underlying asset and size become set, as does the activation date. The only unknown for the contract itself is the strike. However, in terms of pricing the contract, the future price of the underlying asset is also, of course, unknown. Upon activation with its new strike price, the pricing of a forward start option becomes the same as any regular option.

For options on stocks, one benefit is that they enable investors to participate in the total value of the underlying company as the share price increases with dividend growth. In this case, the options holder does not take any market risk before the option becomes active.

Forward start options typically attempt to keep future strike prices at the money or near the money. In this way, the holder will have the right, but not the obligation, to buy or sell the underlying asset in the future but be assured it will initially be at or near the then-current market price. In a sense, employee stock options are a type of forward start option. Here, the company commits to granting at-the-money options to employees without knowing what the stock price will be in the future.

If at the maturity date, the underlying trades below the strike price of the option (for a call) then it expires worthless. If the underlying is above the strike, then the holder exercises it and owns the underlying at the strike price. Typically, as with most options, the holder may sell the option if it is in the money and take the cash." [32]

Ones understood what a forward option is and how it works, we can explain that the database will save the information about the forward start option start and end dates. It will save this information in case we have a contract to buy an amount of assets on a further date, we know that in the start date we might execute the purchase and there will be a currency transaction. However, we can't save this information in derivatives or other assets transactions, because at the moment we register a forward start option, we don't have the asset yet, and maybe we will not execute the option and we will not have the asset, so we will lose this information.

The last information we will store, for each currency transaction, are the transaction author and the transaction owner.

Fund assets transactions

As with currency transaction, in order to ensure that we can make two fund transactions at the same time and store both of them separately, we will assign them different identifiers. This identifier will be a mandatory integer field.

For every fund transaction we need to know with which fund we have done the transaction, so we will store the fund identifier which is the ISIN.

Any transaction with funds involves a currency transaction too. If we buy a participation of a fund, the money has to be taken from a current account, and its balance will decrease. If we sell a participation of a fund, the money we will obtain from this sale will be saved in a current account, and its balance will increase. So, the database will store the IBAN of the current account where the money will be transferred or taken from.

As in any transaction, we will have an account to register the asset we are buying. In the same way, if we are selling an asset this has to be taken from an account. Then a fund transaction will store the funds account related with the transaction, which will be the field called “Depositary & Numbering”

Information we will save in every fund transaction, is the amount of holdings we are buying or selling.

In fund transactions we will also save the settlement price. “The settlement price […] refers to the final price an underlying asset achieves with reference to options contracts to determine whether they are in-the-money (ITM) or out-of-the-money (OTM) at expiration and what their payoffs ought to be. Settlement prices may also be used to compute the net-asset value (NAV) of mutual funds or ETFs on a daily basis.” [33]

Since funds are sold by a merchant entity, in a fund transaction we will have a field to save this entity.

Also, as in currency transactions, in each fund transaction we will save the value, the transaction date, the value date, the quote market, the commission we pay for it, its author and owner

Fixed income securities transactions

As with each type of transaction, the first information we will save in a fixed income transaction is an identifier.

In the same way, we will need to identify the fixed income asset we are buying or selling, so we will save the product code, or its ISIN.

We will need to identify in which values account we will save the fixed income asset we are buying, or where is saved the fixed income asset we are selling. To make these identifications, every fixed income transaction will need a values account identifier, which is the CCV. Also, we need a current account to take the money for the purchase or where to deposit the money in a sale, and for it we will store an IBAN related with the transaction.

Fixed income assets have a nominal value, which is the amount of money that the holder pays for it. At the maturity date, the holder will receive the nominal plus interests, or the nominal plus the last coupon.

When a fixed income assets transaction is made, it is meant to pay a commission, which increases the total paid, so we have to store this information, such the commission paid as the total paid for the transaction.

Sometimes fixed income assets price is given as the asset price plus all coupons that the holder will receive, even so the “ex-coupon price” is the price of the asset where is not included any coupon and we will be able to set it in fixed income transactions.

In fixed income assets, the holder receives, periodically, an amount of money called coupon, which is a percentage of the amount payed for the asset. The amount accrued since last payment is called accrued coupon and the database will be able to save this information in fixed income transactions.

In addition, there are many different types of fixed income assets, and different ways to get all these coupons. Sometimes what we pay for the asset is not the nominal, so we can pay for the nominal minus all coupons we will receive in the future. To save this information too, we will have a field called “total cash” which will be the amount of money we are paying for the asset and the commissions. In case we pay for all nominal value, this field will have the same value.

Sometimes a fixed income transaction needs a broker to be executed, and our database will be able to save the broker who has executed the transaction.

“A broker is an individual or firm that charges a fee or commission for executing buy and sell orders submitted by an investor. A broker also refers to the role of a firm when it acts as an agent for a customer and charges the customer a commission for its services.

As well as executing client orders, brokers may provide investors with research, investment plans and market intelligence. They may also cross-sell other financial products and services their brokerage firm offers, such as access to a private client offering that provides tailored solutions to high net worth clients. In the past, only the wealthy could afford a broker and access the stock market. Online broking triggered an explosion of discount brokers, which allow investors to trade at a lower cost, but without personalized advice.” [34]

Variable income securities transactions

As in every financial asset transaction we will save a transaction identifier, the product code, the IBAN related with the transaction, its date, the value date, the quote market of the asset, the broker if necessary, commissions, the author and the owner of the transaction.

Fixed income assets transactions in these transactions we will save the values account identifier where the shares will be saved, or sold from. This identifier is again the CCV.

Obviously in every transaction with variable income assets we will save the number of assets we are buying or selling and their price. We also will save the total paid, which will be the result of multiply the price by the number of shares plus the commissions.

We have to notice that when we try to buy or sell more than one share at the market price, it will be so difficult to buy/sell all of them at the same price. So, when this happens we will have a transaction for each different price we are achieving to buy/sell these shares.

Derivative assets transactions

As in every financial asset transaction, with every derivative assets transaction we will save a transaction identifier, the product identifier, the IBAN related with it, the amount

of titles we are buying or selling, the transaction date, the value date, quote market, the broker if it is needed, the total cash, the price of the asset, commissions, the total paid, the author and the owner.

However, we have some information that only appears in the financial assets account where the asset will be stored in a purchase, or taken from in a sale. This information will be the maturity and the asset and the underlying amount.

The maturity of the asset is the date when the asset will be sold.

The underlying amount. "Underlying applies to both equities and derivatives. In derivatives, underlying refers to the security that must be delivered when a derivative contract, such as a put or call option, is exercised.

There are two main types of investments: debt and equity. Debt must be paid back and investors are compensated in the form of interest payments. Equity is not required to be paid back and investors are compensated by share price appreciation or dividends. Both of these investments have specific cash flows and benefits depending on the individual investor." [35]

Properties transactions

As we have seen in the previous assets in properties, or real estate assets, the information to save in the database about transactions made with them will be a transaction identifier, a product identifier that will be the land register code, the IBAN related, the transaction date, the price of the property, the transaction author and the transaction owner.

In addition, in each property transaction we will save the reference register ID. This ID is what will identify the account where a property bought will be registered, or where a property sold will be taken from.

2.6 Other information

Leading up to this section we have defined all information we will need to store, regarding the different financial instruments, accounts where store them, transactions that we can make with them and all their attributes. But there is more information we need to save, and we will make a difference between additional instruments information and user information.

2.6.1 Additional instruments information

The price of most of the instruments is variable, it is volatile, and they have a price when we are buying it, but when we want to sell it, it's usually because its price has changed. If we want to have the capability to give the chance to the final user, to make some reports about profitability of her/his assets, we will have to store many prices. This means a lot of information to save, and if we store this information in the same table where the asset is stored, we will have a lot of information duplicated, so for each row with a new price we will store a new row of attributes that never change.

At the same time there are many quote markets where to buy financial assets, even we can find shares from the same company in different markets. In addition, markets have a lot of information, so in the same way as prices, it is not worth it to store all this information in the same table as the object or the transaction, so we will also build a table to store all information about different markets.

Let we see which information we need to store in our database.

Prices

- ISIN/ID: first of all, we will need to identify the asset for which we are saving the price, and this attribute is the asset identification code.
- Market: this table will save the market where is set this price. Sometimes, in different markets, the same asset can have different prices.
- Date: this is important information since many financial assets prices are changing every second.
- Price: the price will be stored without the currency on what it is expressed, It is because the market related has this information.

Once we have this information, we will be able to store all prices that we want to be able to know.

Markets

Markets are so complex, and they have many attributes, which are:

- Market id: it is the market identifier which is a string wrote in capital letters.
- County: it is the country where the market is developed.
- Currency: it is the currency in which the market works.
- Settlement currency: is the currency in which the money from a transaction is paid. In this way, a settlement currency can differ from a truncation currency. "The transaction currency multiplied by the exchange rate will indicate the settlement currency amount for the transaction. " [36]
- Market description: this is the default description of every financial market.
- ISO code: some markets has an ISO code, so in our database we will be able to store it, although it will not be mandatory since not every market has this code.
- International code: it is the international code of every market. It is a string and sometimes is the name of the country, wrote in English, where the market is developed.
- Derivatives market: this is a Boolean attribute that, when its value is true, means it is a derivatives market, and if its value is false it means that it is not a derivatives market.
- Market type: although we can classify the financial markets for many different characteristics, this attribute will have to store the information about if it is an official stock exchange market, if it is another organized market, if its OTC ("Over-the-counter (OTC) refers to the process of how securities are traded for companies that are not listed on a formal exchange[…].Securities that are traded over-the-counter are traded via a broker-dealer network as opposed to on a centralized exchange. These securities do not meet the requirements to have a listing on a standard market exchange. " [37]), or a not quoted market.
- Guarantee Percentage: "A financial guarantee is a non-cancellable indemnity bond backed by an insurer to guarantee investors that principal and interest payments will be made." [38]

- Closing time: it is the time, in the country's capital city, at which the market closes.
- Rate: some markets has a rate, or commissions, to work with them. This field will say which rate is used in the market.
- CNMV: it will be a short description that the CNMV (Comisión Naciónal del Mercado de Valores) makes of every market.
- Execution cost: "The difference between the execution price of a security and the price that would have existed in the absence of a trade" [39]
- Bloomberg code: this is the code that Bloomberg LP gives to markets. Bloomberg is an American company that its defined by itself "As a global information and technology company, we connect decision makers to a dynamic network of data, people and ideas – accurately delivering business and financial information, news and insights to customers around the world." [40]
- BIC code: some markets have this a BIC code, but not all since it is a banking code. "This international banking code that allows a unique identification of each credit institution and / or its offices". [41]
- Regulated: it is a Boolean attribute which will mean that the market is regulated by an authority, when its value is true. If its value is false, it means the market is not regulated by any authority
- Organized: it is a Boolean attribute which will mean that the market is organized, when its value is true. If its value is false, it means that it's not. An organized market is "A formal market in a specific place in which buyers and sellers meet to trade according to agreed rules and procedures. Stock exchanges, financial futures exchanges, and commodity markets are examples of organized markets." [42]

Transaction types

To be able to do a proper analysis of our transactions, know how much taxes we have to pay at the end of the year, how much we have paid in our properties, how much we have paid in bills, or any analysis that we would like to do, we need to classify every movement we make in our current accounts. That why we will store the following information:

- Name: it is a short name, with only 4 characters, which will be the identifier.

- Family affected: it is a Boolean attribute. When its value is true, it means that the movement affects to whole family or organization, when its value is false it means that the transaction only affects to one person.
- Professional affected: it is a Boolean attribute whose meaning, when it is true, the transaction is originated for a professional purpose. If its value is false, it will mean that the movement doesn't have a professional purpose.
- Other accounts affected: it is another Boolean attribute whose meaning when its value is true, will mean that this type affects more accounts, a part of the current account which is related with the transaction which is pointing to this transaction type. If the value is false will mean that the transaction is only affecting the current account.
- Tax repercussion: this is also a Boolean field. When its value is true, it means that the movement can have a repercussion, when we will be calculating the taxes we have to pay at the end of the year. If its value is false it will mean that the movement have no repercussion on taxes.
- Description: this is a field to have a little description about the transaction types.

Banks

Accounts are opened in banking entities, and all information we will store about banking entities are the following:

- Bank ID
- Bank name
- Address

2.6.2 User information

Our database will be built to let an app run on it and this app should be able to have more than one account, or manage financial assets of more than one person, for example a family.

In addition, it would have no sense to store information about financial instruments, transactions and accounts but not the owner.

The future application must be able to calculate fiscal taxes that the user will have to pay based on its earnings with investments and financial products.

Transactions have attributes as author and owner of them, so we will need to store information about the user, and this is the following information.

Subject

The subject may be a person or an organization, like a company, a foundation or an association.

The database will be able to store the following information about a subject:

- NIF/NIE/: this is the fiscal identification, always needed to buy a financial asset. As it has said before this field must accept NIF and CIF codes. They have different formats, but both have nine characters in total.
- Name: the name of the person or organization.
- Fiscal address: it is where the subject is legally registered.
- Postal/Physical address: it is the address where the subject use to receive postal mail. It can be the same as the fiscal address.
- Phone number
- Email
- Password: the database will be able to store a password to be used for an account in the application. The password must be saved encrypted. The responsibility of it is not going to be the database, so we think all security layer has to be done in the application layer.
- Unsubscribe date: when a user has an unsubscribe date, we can say that her/his account is not used at the moment.

This information is the same for people and companies, but there is some information that a person can have and a company can't and vice versa. So now it will be enumerated the information that the database will have to store about a person:

- 1st and 2nd Surnames
- Birth date
- Civil status: it will be stored because it has an effect on fiscal taxes.

- Spouse: this is the NIF/NIE of the spouse that may be stored in the database as well, as another subject.
- Children: this is the number of children that the subject have. It will be used to calculate fiscal taxes too.

In the following list there is the information that the database will have to store about an organization:

- Sector: as the financial instruments attribute that have the same name, we will store in which sector the organization is developed.
- Creation/Foundation date

An address has lots of information, and many people can have the same fiscal and postal address, we need to enumerate and understand all of it.

The next list is all information that an address gives us, and which the database will have to store.

- Road type: like avenue, road or street. We can find it in its name.
- Road/street name
- Township
- Postal code
- City
- Country
- State/Province: depending on the country we can have both, so it will be a long string field, in the database, to be able to store both in it.
- Number
- Floor
- Door
- Km/mile

Participation

In order to store the information about shared accounts between more than one subjects, we need to store the participation of each one in the account. This will help the user to calculate taxes that she/he has to pay at the end of the year.

3. The database

3.1 Basic concepts

In this section, we will first define the meaning of some database concepts, since we will talk about them through this section.

Primary key

"The primary key constraint uniquely identifies each record in a table. Primary keys must contain unique values, and cannot contain null values. A table can have only one primary key; and in the table, this primary key can consist of single or multiple columns (fields)." [43]

Foreign key

"A foreign hey is a key used to link two tables together. A foreign hey is a field (or collection of fields) in one table that refers to the primary key in another table. The table containing the foreign key is called the child table, and the table containing the candidate key is called the referenced or parent table." [44]

Composite key

"A composite key is a combination of two or more columns in a table that can be used to uniquely identify each row in the table when the columns are combined uniqueness is guaranteed, but when it taken individually it does not guarantee uniqueness.

Sometimes more than one attributes are needed to uniquely identify an entity. A primary key that is made by the combination of more than one attribute is known as a composite key." [45]

3.2 Building tables and relationships

In the previous sections we have seen all information we have to store in our database, but now we have to structure this information and build a consistent database, where the information is not mixed, and where the information is duplicated as less as possible.

First of all we have to realize that in the information, that we have identified, there are many attributes which are repeated in many similar entities, like the subscribe date in every account, the name in person and organization, and product type in instruments. So what we will do, to decrease the number of columns in the whole database, is build tables with common attributes in accounts, subjects and instruments.

3.2.1 Software used

As a recommendation of an involved person in the project, to develop the database, it has been used the software called MySQL Wotkbench, and as a server to build the database in, Oracle GlassFish server.

"MySQL Workbench is a unified visual tool for database architects, developers, and DBAs. MySQL Workbench provides data modelling, SQL development, and comprehensive administration tools for server configuration, user administration, backup, and much more.

MySQL Workbench enables a DBA, developer, or data architect to visually design, model, generate, and manage databases. It includes everything a data modeller needs for creating complex ER models, forward and reverse engineering, and also delivers key features for performing difficult change management and documentation tasks […]." [46]

"Oracle GlassFish Server is the world's first implementation of the Java Platform, Enterprise Edition (Java EE) 6 specification. Built using the GlassFish Server Open Source Edition, Oracle GlassFish Server delivers a flexible, lightweight, and production-ready Java EE 6 application server." [47]

3.2.2 Financial instruments

Tables

It is not always the best decision to build a table with the common attributes as with instruments. These entities, once we will have them stored in the database, at most we will need to update registers, and we will not save each instrument more than one time. In addition, not all of them have the same type of identifiers, so we would have to put an additional attribute as an identifier to relate all of them to the table that would have the common attributes.

Also we will have to read the instruments attributes several times, so it will be quicker and more efficient to have a single table query.

An observation we must do before start defining financial instrument tables is that, to make a difference between instruments ant transactions, every instrument table name will be called the name of the instrument it represents, plus the word "Object". So the instruments tables will be the following.

CurrencyObject (Code, Name, Value, Issuing_entity, Sector, Instrument_class, *Product_type*, Quoted, Issuing_date, Counterpart)

FundObject (AssetCode, Code, *Quote_market*, Name, Value, Issuing_date, Sector, Instrument_class, *Product_type*, Quoted, Issuing_date, *Currency*, Managing_entity, Composition)

Fixed_incomeObject (AssetCode, Code, *Quote_market*, Name, Value, Issuing_entity, Sector, Instrument_class, *Product_type*, Quoted, *Currency*, Issuing_date, Accrual_date, First_cupon_date, Expiration_date, Nominal, Interest_rate, Payday, Days_of_calculation, Nominal_reduction, Counterpart, Physic_title, First_call, PUT)

ProperyObject (Code, Name, Value, Issuing_entity, Sector, Instrument_class, *Product_type*, *Currency*, Issuing_date, LandRegistry_value)

DerivativeObject (AssetCode, Code, *Quote_market*, Name, Value, Issuing_entity, Sector, Instrument_class, *Product_type*, Quoted, *Currency*, Issuing_date, Expiration_date, Nominal)

Variable_incomeObject (AssetCode, Code, *Quote_market*, Name, Value, Issuing_entity, Sector, Instrument_class, *Product_type*, Quoted, *Currency*, Issuing_date, Nominal, Physic_title)

Product_type (ProductType_ID, Description1, Description2, Description3)

Market (Maret, Country, *Currency*, Settlement_Currency, Market_description, ISO_Code10989MIC, International_Code, Derivatives_Market, Market_Type, Guarantee_Percentage, Closing_Time, Fee, CNMV, Execution_Cost, Bloomberg_Code, BIC_Code, Regulated, Organized)

In all these tables we can see that there are some tables whose primary keys is an "AssetCode". This happens because sometimes we can find the same asset code in more than one quote market, and even though the code is the same, maybe because the product are shares of the same company, products have to be differentiated if we buy them in different quote markets. Then, to identify the correct object when we store prices or transactions, we only need to make a reference to this "AssetCode".

Relationships

Even the "*Quote_market*" is a part of the primary key of these instruments, it is also a foreign key. It means all financial instruments, except "PropertyObject", will point "Market" table.

In all these instruments we can see that the "*Product_type*" is a foreign key, because we have built a table called "*Product_type*" that will allow the user to tag assets as a product type, independently of its instrument class, and all instrument table's rows will point to that table which will define all different types we want to have.

We can see that in each instrument, except in "CurrencyObject", there is the attribute "*Currency*" as a foreign key. This attribute will refer to the "CurrencyObject" identifier that is its attribute "Code". So every attribute will point to "CurrencyObject" too.

Let's see three diagrams to understand better these relationships between financial instruments, "Product_type" and "Market" tables.

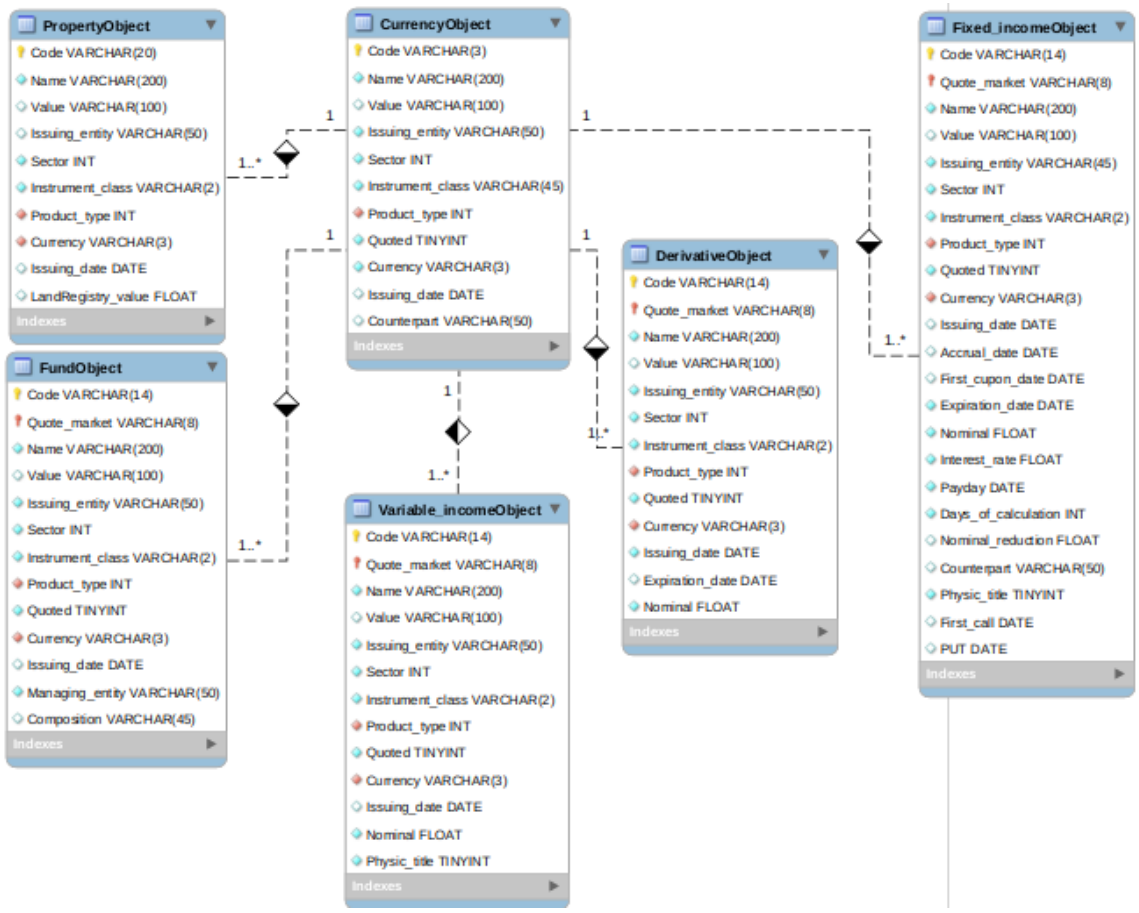

Figure 1: Financial instruments tables UML relationships

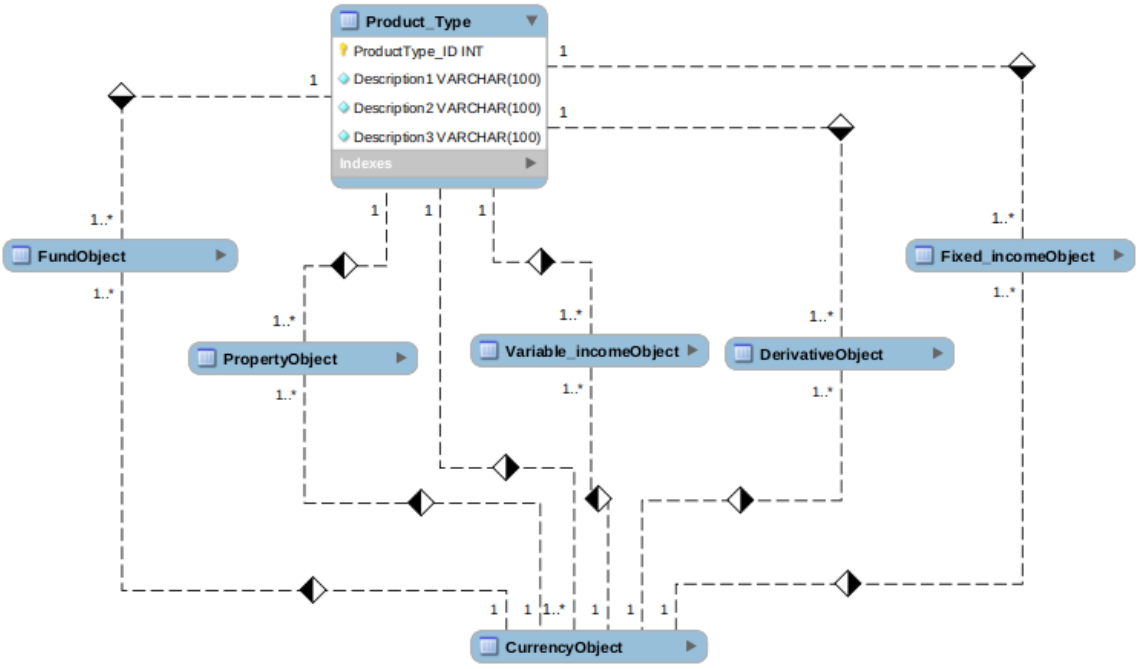

Figure 2: Financial instruments tables and Product type table UML relationships

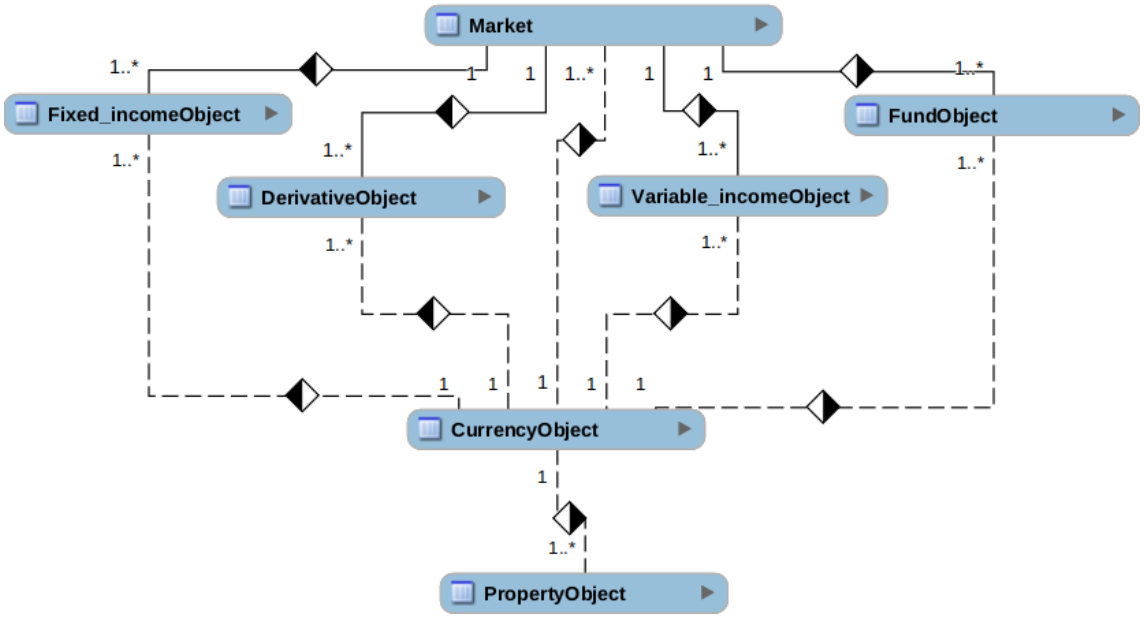

Figure 3: Financial instruments tables and Market table UML relationships

3.2.3 Accounts

Before defining account tables, we must understand how the database will work between accounts, assets and transactions.

Even we talk about assets allocated in accounts, because it is the way we use to understand accounts, as a drawer with assets in it, the database will not work as it.

In our database, we will have transactions that will be pointing to the assets with which these transactions have been made. Then these transactions will be the entities which will be related with accounts.

Tables

In every account we will have the sign up date, the unsubscribe date, and the account type.

In addition, when we have to build the relationship between users and accounts, the code that we have to write, and the number of relationships we will have to control in the future, will requires less work if we build a single table where we can find all accounts.

So we will build a table called “Account”, which will only have these attributes and every different account will have to have a foreign key which will refers to the Account’s primary key.

The tables we will build in order to model accounts will be the following:

Account (*Numbering*, Account_type, Sing_up_date, Unsubscribe_date)

CurrentAccount (IBAN, Balance, *Currency*, *Banking_entity*, *Account_numFK*)

Financial_assetsAccount (Numbering, ISIN, Balance, *Banking_entity*, *Account_numFKfinantial*)

ParticipantAccount (Depositary_and_Numbering, Balance, *Banking_entity*, *Account_num_FKparticipant*)

ReferenceRegister (ReferenceRegister_ID, *Account_numFKreference*)

ValuesAccount (ValuesAccountCode, CCV, ISIN, Balance, *Bank_entity*, *Account_num_FKvalues*)

Banks (BankID, Bank_Name, *Address*)

In some accounts we can see an attribute called “ISIN”, which is not a proper account attribute, but since some accounts can only be related with an asset we will store its ISIN in the table. We can, also, see that it’s not a foreign key. This is made to avoid more unnecessary circular references between tables. Even the ISIN in this table is wrong, the account will be related with the correct asset with the transactions related with the account.

Something that can seem that has no sense, is to have the table “ReferenceRegister, because we are only storing the “ReferenceRegister_ID” which is the same identifier as the property. However there is a reason to build this table, it’s the way we have designed the database and the relationship between accounts, assets and transactions. So to keep consistency we have built this table.

Relationships

Since accounts can have different types of identifiers, or codes, we will need an extra code which will be the foreign key in each type of account, to link them to the "Account" table.

Also "*Banking_entity*" is a foreign key that appears in almost every account. That is because we will have another table with all information about banking entities.

Let's see an UML diagram to have a better idea how account foreign keys make these relations between accounts, "CurrencyObject", and "Banks" tables.

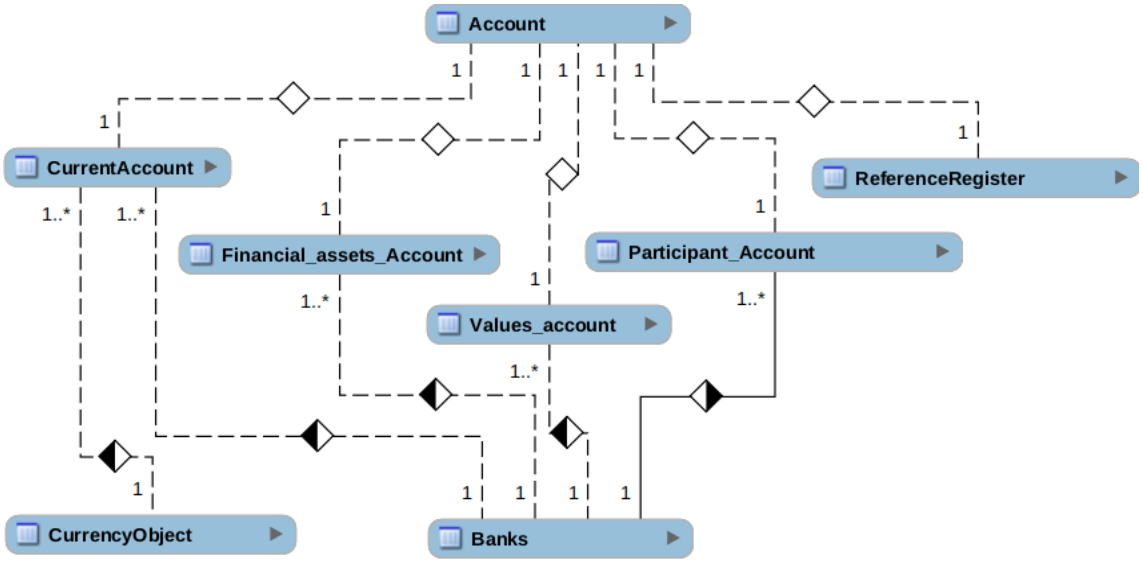

Figure 4: Accounts UML relationships

3.2.4 Subjects

Subjects are an example that more than one entity type have the same type of identifier, all of them have 9 characters. The first character can be a letter or not, the next 7 always are numbers and the last one is always a letter. In addition they have many common attributes, so in the database we will build a table with these common attributes and a table for each type of subjects, person and organization.

Tables

To store all the information about subjects, a single table isn't enough, so we will build five different tables. Let's see them.

Subject (NIF, Name, *Postal_Address_Id*, *Fisical_Address_Id*, Phone_number, Email, ClientPassword, Unsubscrive_date)

Person (*NIF*, 1stSurname, 2ndSurname, BirthDate, Civil_status, Spouse, Children)

Organization (*NIF*, Sector, CreationDate)

Participation (*Subject_NIF*, *Account_Numbering*, Participation)

Address (Address_Id, Road_type, Road_Name, PostalCode, City, State/Province, Country, Number, Floor, Door)

Relationships

In these tables, NIF is the primary key in the parent table "Subject", but at the same time it also is a foreign key in both child tables "Person" and "Organization".

However, the relationship between a user and her/his addresses is made in the table Subject, which has two foreign keys, "*Postal_address*" and "*Fiscal_address*".

To have a table where we can find every single account the user has, independently of the account type, will help us building only a relationship between users and accounts.

Let's see an UML diagram to understand better these relationships.

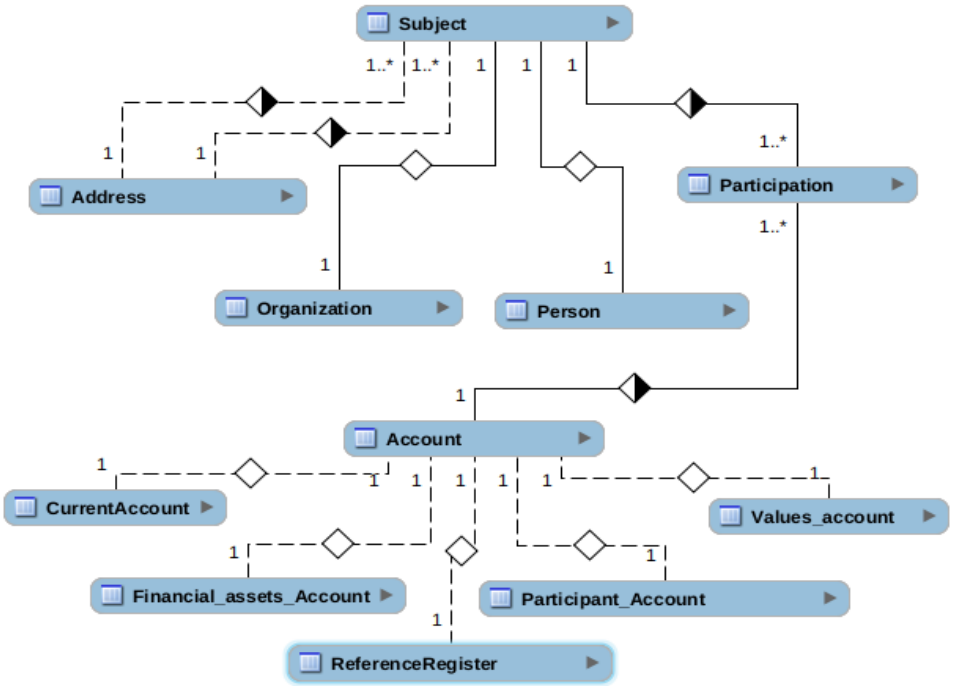

Figure 5: Subjects, addresses, participation and accounts UML relationships

3.2.5 Transactions

Transactions are the tables where every transaction made with the financial assets will be stored, and all of them will be related with an account.

With all information we have seen we have to store, let's we see how we have built the transactions tables.

Tables

CurrencyTransactions (Movement_Id, *IBAN_Account_Related*, *IBAN_Account_related2*, *Currency1*, *Currency2*, Amount, *Price*, Commissions, Cash_transaction, Transaction_date, Value_date, Start_forward_date, Final_forward_date, *Transaction_author*, *Transaction_owner*, Transaction_concept, Transaction_Origin, *Transaction_type*)

Fixed_IncomeTransactions (Transaction_code, *ISINrelated*, *IBAN_Related*, *CCV_Related*, Amount, Transaction_date, Date_value, *Nominal*, Broker, Ex_cupon_price, Run_cupon, Total_cash, Comissions, Total_paid, *Transaction_author*, *Transaction_owner*)

Variable_IncomeTransactions (Transaction_code, *ISINrelated*, *IBAN_Related*, *CCV_Related*, Transaction_date, Date_value, Number_of_shares, Broker, *Price*, Comissions, Total_paid, *Transaction_author*, *Transaction_owner*)

FundTransactions (Transaction_Code, *Fund_code*, *IBAN_Related*, *Depositary_and_NumberingRelated*, Number_of_holdings, Value, Transaction_date, *Settlement_price*, Merchant_entity, *Transaction_author*, *Transaction_owner*)

DerivativeTransactions (Transaction_code, *Product_code*, *IBAN_Related*, *FA_Account_Related*, *Price*, Titles_amount, Maturity_of_the_asset, Transaction_date, Value_date, Broker, Underluing_amount, Total_cash, Commisions, Total_paid, *Transaction_author*, *Transaction_owner*)

ProperyTransactions (Transacion_code, *Land_Registry_code*, *IBAN_related*, *ReferenceRegister_Id*, Transaction_date, *Price*, *Transaction_author*, *Transaction_owner*)

Prices (Price_Code, AssetCode, *Quote_market*, Date, Price, *Currency*)

We should notice that, even if transactions have an attribute that indicates the quote market where we are buying or selling an asset, we do not have the attribute in these tables. It is because the quote market where an asset is bought is already stored in the asset, in the tables that are modelling financial assets. So if we don't include the quote market in transactions tables we are avoiding another unnecessary circular reference between financial instruments, transactions and markets.

Relationships

To make sure we are linking every field with the correct account, we are not using the parent table "Account" to relate transactions with accounts. Instead of that we will link every transaction directly with each account table, child tables, which must be related with.

Let's see some UML diagrams to understand better these relationships between transactions tables and their related tables.

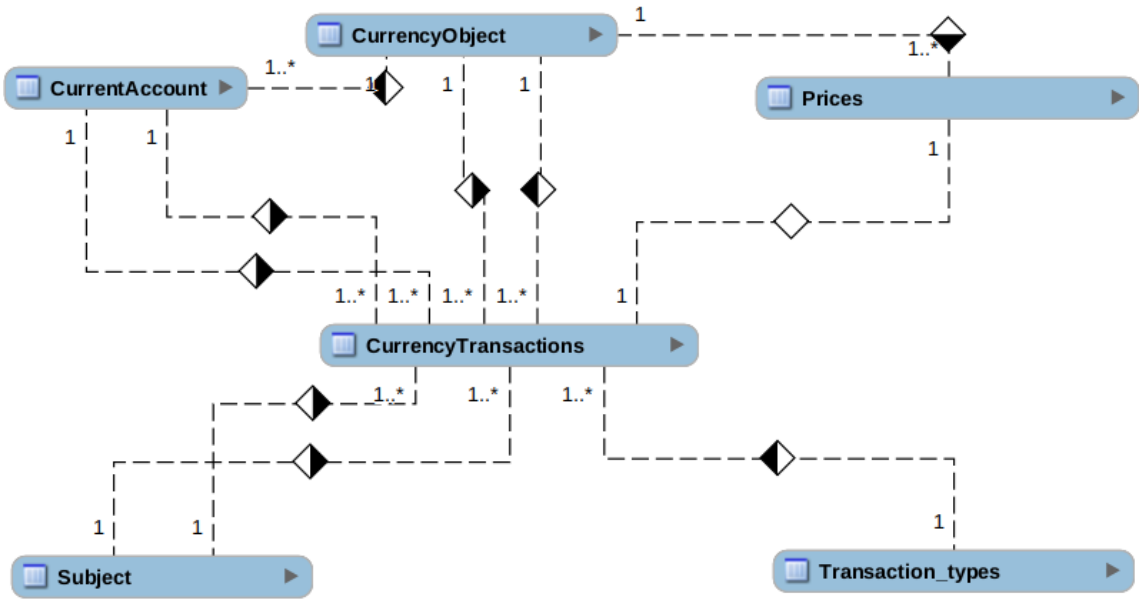

Figure 6: Currency transactions table relationships

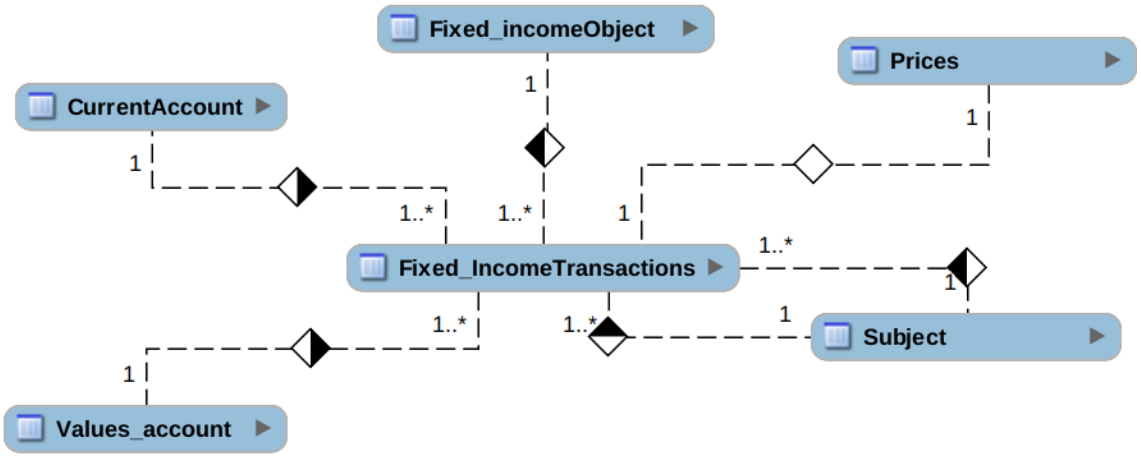

Figure 7: Fixed income transactions UML relationships

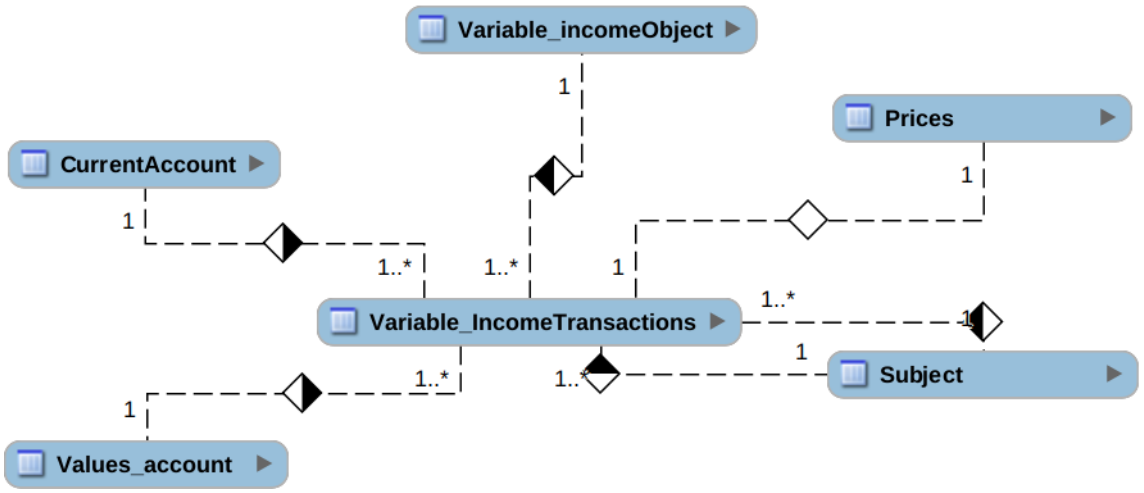

Figure 8: Variable income transactions UML relationships

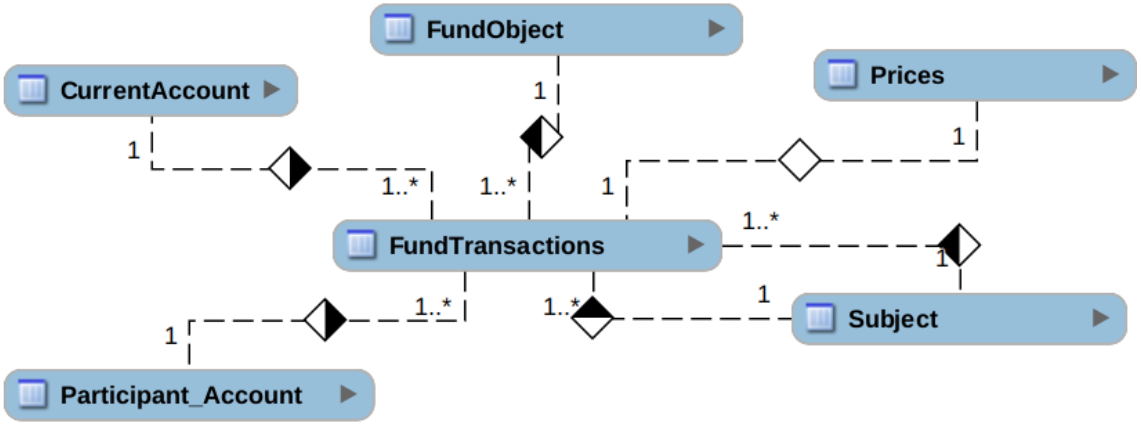

Figure 9: Fund transactions UML relationships

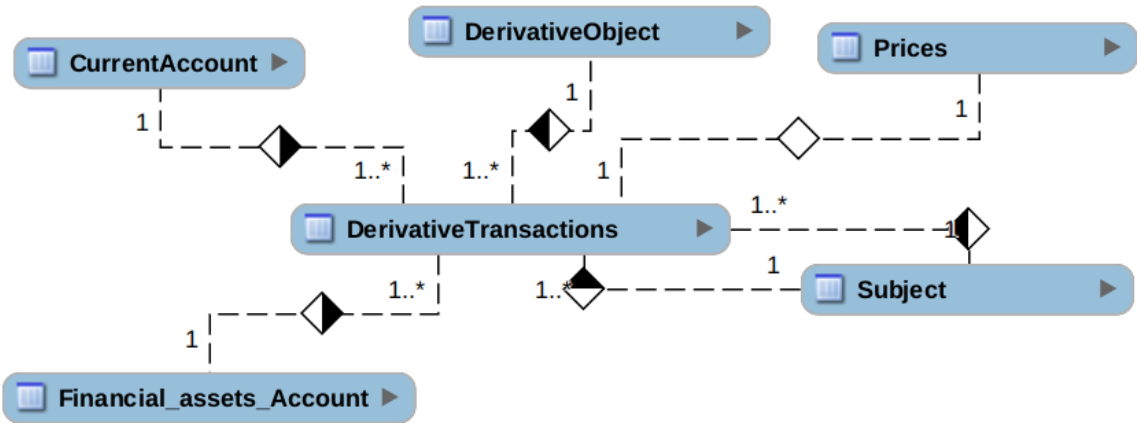

Figure 10: Derivative transactions UML relationships

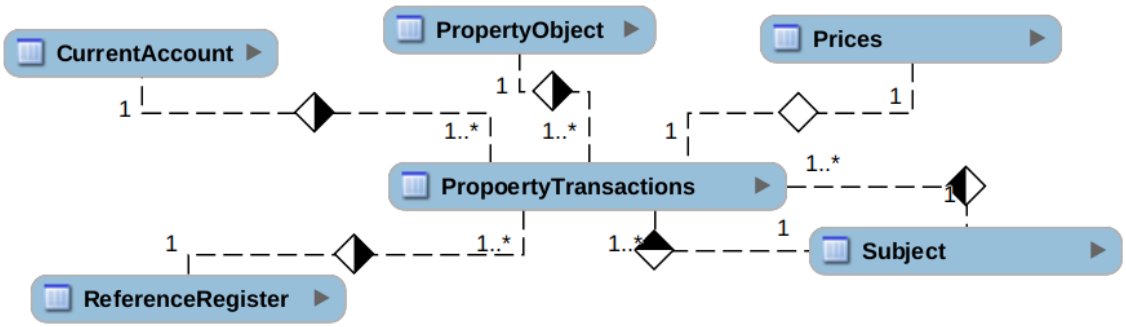

Figure 11: Properties transactions UML relationships

3.3 Global view

In the previous section we have been seeing pictures about some tables and their relationships. Now we should see all tables and relationships all together in a single picture, to have a better idea about how complex is the database in terms of number of tables and relationships

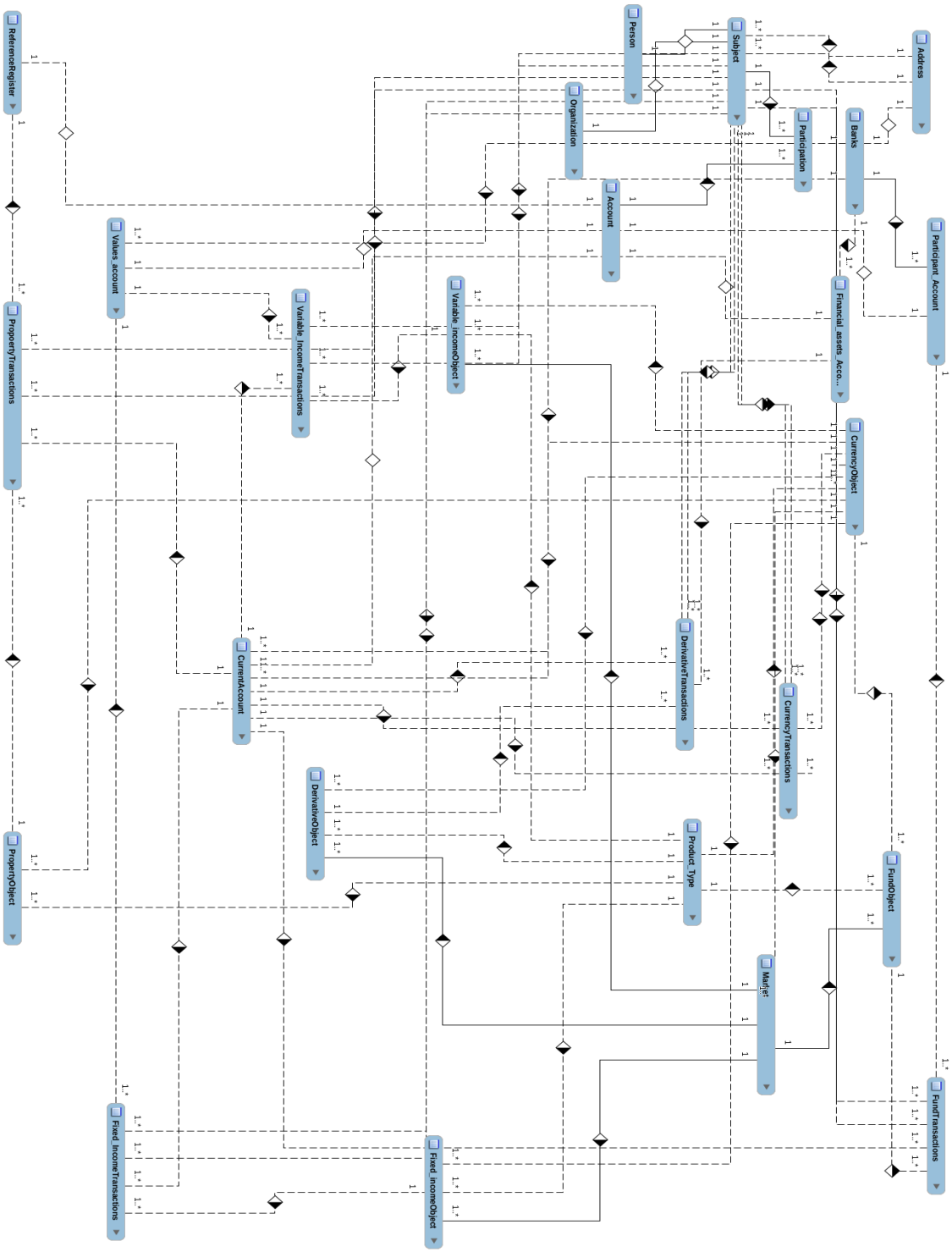

Figure 12: Global view of all tables and relationships in the database

3.4 Triggers needed

"A SQL trigger is a special type of stored procedure. It is special because it is not called directly like a stored procedure. The main difference between a trigger and a stored procedure is that a trigger is called automatically when a data modification event is made against a table whereas a stored procedure must be called explicitly." [46]

An example, in our database a trigger could be useful to increase or decrease the balance in the current account related with the transaction. However, in the transaction we don't store the price, which is stored in another table related with the transaction table. In addition, the prices table has a currency that has to match with the current account currency.

In the database layer we can't make all these checks or, at least, it is easier and quicker if all these checks are done in the app layer. This is the reason why we have decided to build only three triggers.

These triggers will help the app in any asset transaction anyway. Since each asset transaction causes a currency transaction, and we know that a currency transaction will always be always made with the same currency that this transaction is made, and also the amount of the transaction will be the amount that the current account will increase or decrease, we can modify the balance of a current account after a currency transaction.

So we will define a trigger that will be shot up after a CurrencyTransactions insert query. It will update the current account balance.

In the same way, we will define a trigger that will be shot up after a CurrencyTransactions delete query. It will also update the current account balance, but in the opposite sign that the transaction amount is.

We will also have a trigger before CurrencyTransactions update query. This will take the current amount at the current account minus the old value of the amount of the transaction we are updating plus the new amount of that transaction. If the update is not because of a change in the amount of the transaction, the current account value will not variate.

3.5 Database creation SQL code

We can find all database creation SQL code at the Annex 1

4. Testing the database

In computer science, in order to build a software, database or any product, we test it.

This is so useful to find issues and bad developed features, so you can change whatever is not correct before throw the product to production.

In our case, the test has to make sure that we can build the database and execute insert, update, read, and delete queries in every single table, and also that the trigger is working as it is expected.

This could be easy to do, building the database in our computer and try to execute all queries we need to test in each table using a simple terminal.

However, not everyone involved in the project is used to work on terminals or with SQL, so not everyone has the knowledge to execute SQL queries. So, we decided to make a simple web application to interact with the database. Not all the involved people are living in the same city. So, in order to let everyone test the database, we had decided to have a laptop constantly connected to the Internet, with a TeamViewer running and also an Ubuntu virtual machine where we have built the database and the testing web application.

The application has been developed in java, with the NetBeans framework, and built in a MySQL server, called GlasFish server.

Since the application wasn't a part of this project, once we have had the database built in the server, we built the application using java Faces servlet.

"JavaServer Faces (JSF) is the Java standard technology for building component-based, event-oriented web interfaces. Like JavaServer Pages (JSP), JSF allows access to server-side data and logic. Unlike JSP, which is essentially an HTML page imbued with server-side capabilities, JSF is an XML document that represents formal components in a logical tree. JSF components are backed by Java objects, which are independent of the HTML and have the full range of Java abilities, including accessing remote APIs and databases.

The key idea to a framework like JSF is to encapsulate (or *wrap*) client-side technologies like HTML, CSS, and JavaScript, allowing developers to build web interfaces without much interaction with these technologies." [47]

The testing application is so useful to interact with the database without any knowledge of SQL language. It gives a friendly and simple interface to see a tables index. Each table in the index is a link to see all data stored in the table you have clicked on. To execute the insert and update queries in the interface is so easy to use), because it gives you fields to fill, with labels to identify what it is meant to be in each field. At the same time the application gives you links to delete, update or see a row of information, and also go from a table view to the tables index, as we can see in next pictures.

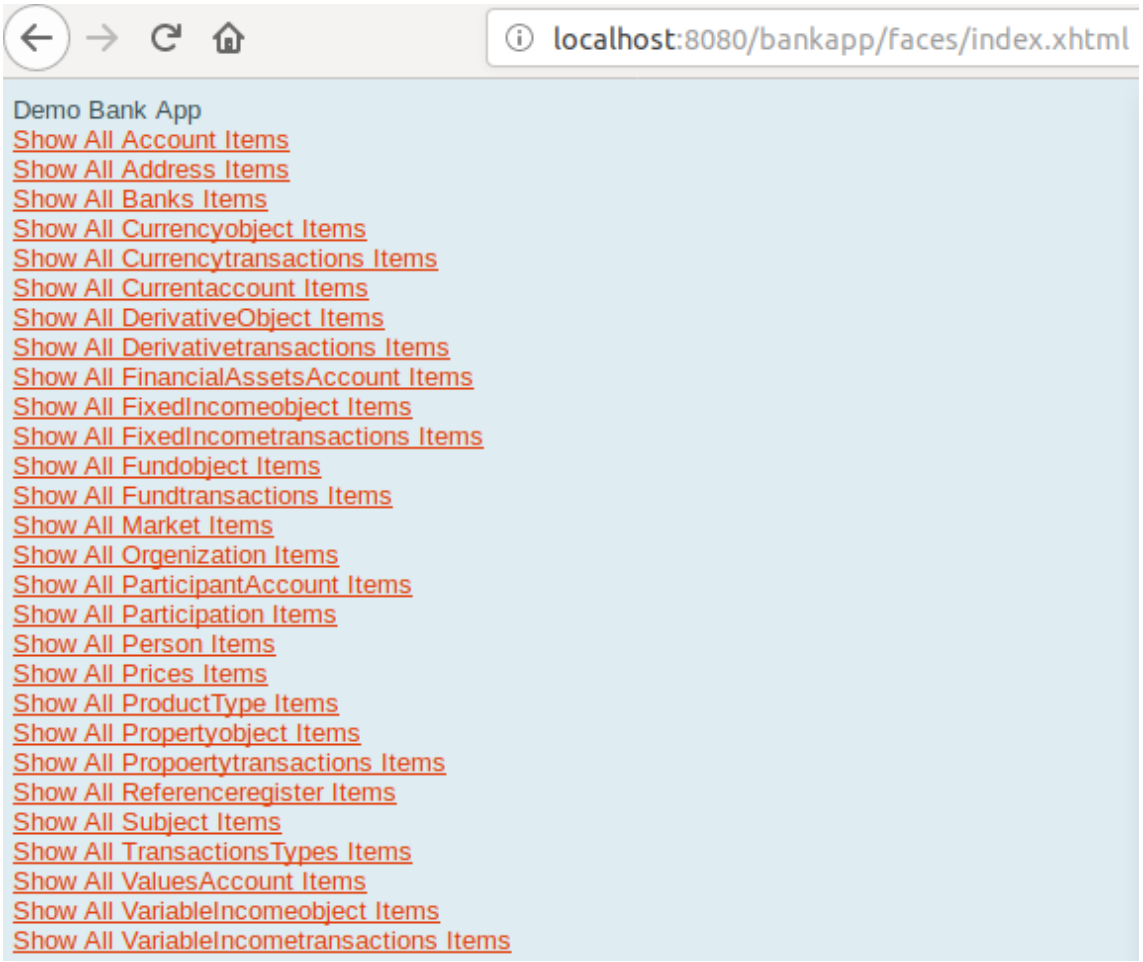

Figure 13: Application index view

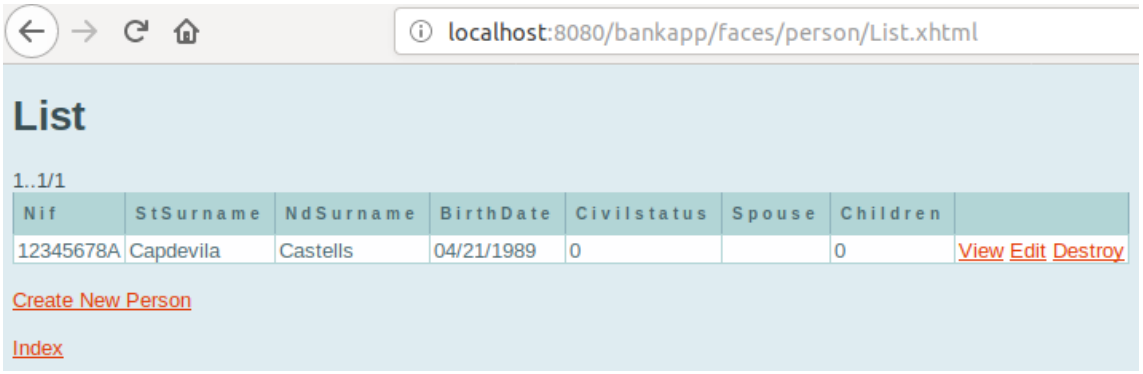

Figure 14: Person table list view

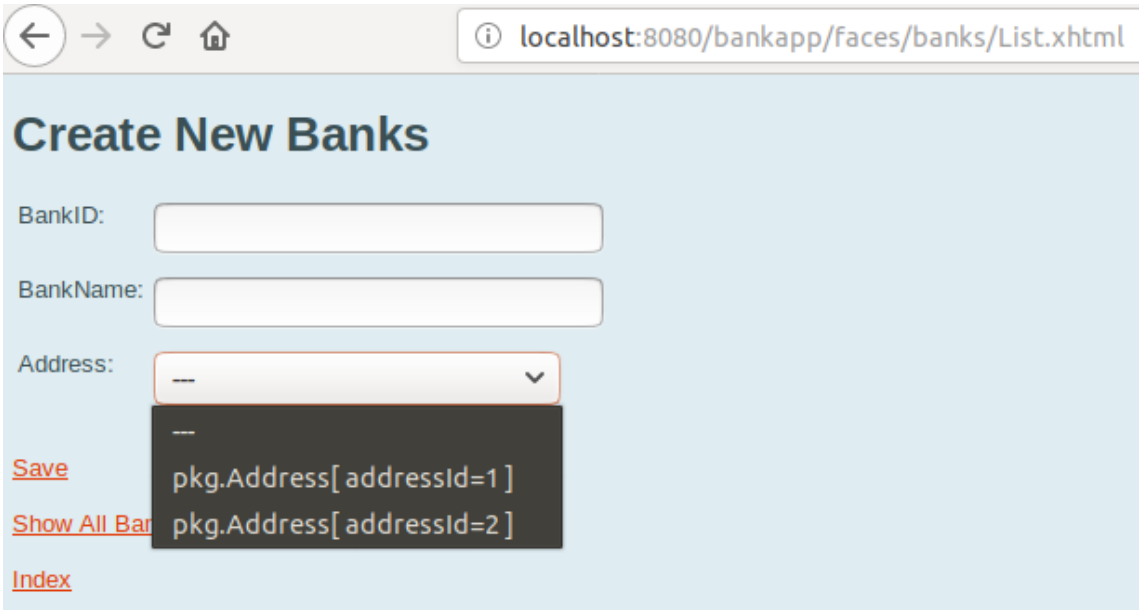

Figure 15: Bank creation view

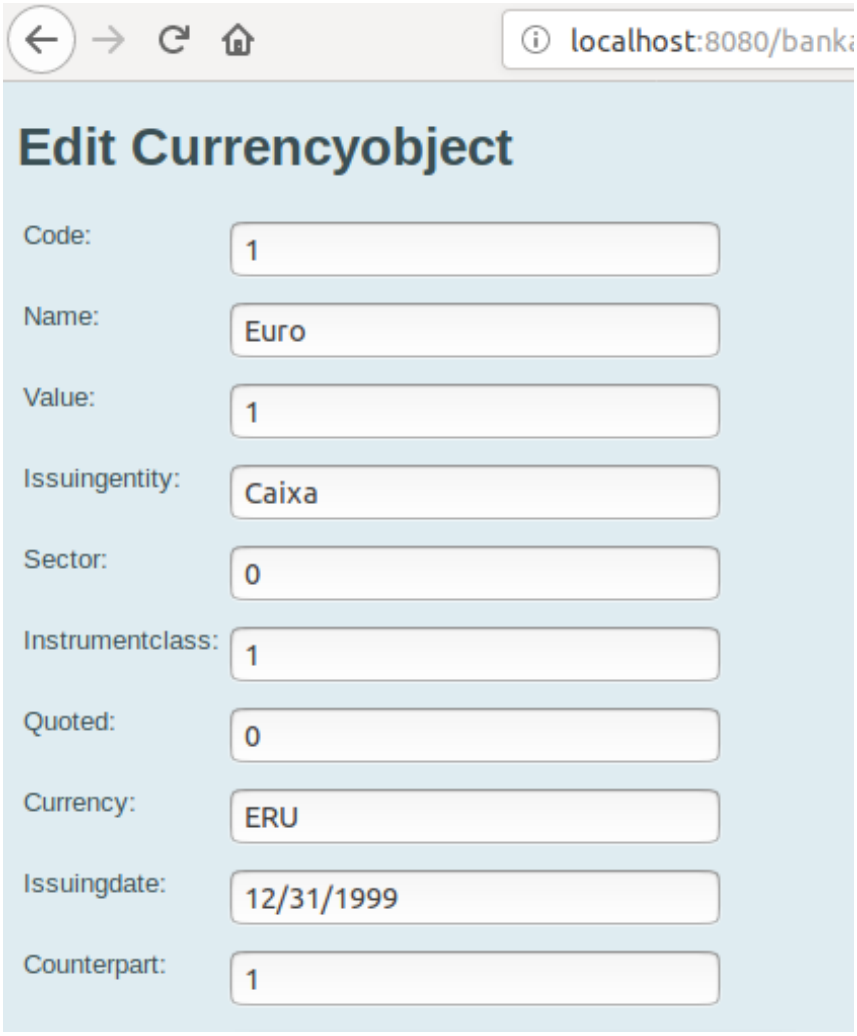

Figure 16: CurrencyObject edit view

Testing the database we could find some typing mistakes in tables names and attributes. Also, we had found some mistakes with foreign keys and relationships between tables and, at the same time we were doing all changes to fix issues and mistakes in the database, we was reflecting these changes on the SQL code which will build the database, and also in the testing application.

Making changes on the database and on the SQL code has been easy and quick, but with that we have found an expensive work, in terms of time and number of files to change, in the application.

To understand why a little change in the database became a big number of changes in the application we need to understand how it works.

Java Faces builds three java classes for each table. One of them is the class whose attributes are all attributes of the table, and two more to interact with the xhtml when it needs to show a list of attributes of a class, to delete a row of a table or to link two rows from different tables. In addition, if the table has a composite primary key, an additional java class is created, in order to have a single attribute primary key in the main class, which is an instance of the new class created.

Then, for each table there are four xhtml files, one for each different web page that we can see in the application. A file for the create page, one for the edit page, one for the list page which gives us the view of all table and all rows stored in it, and another for the view page that gives us another view for the row selected.

Also, there is another file where there are all the labels that will be shown in the application, when the xhtml files are calling them.

So, when we have had to delete a single typing mistake in an attribute that was not a foreign key and it was not referenced by another table, we have had to execute a SQL query to delete the attribute in the database, delete it in the SQL code that builds the database, and delete this attribute in, at least, five files in the web application.

If the change was a typing mistake in a composite foreign key which, at the same time was referenced by another table's attribute, the work on the database was some queries to correct the mistake on the name and all references, and also the same on SQL code. But when we have to reflect all changes on the app, the time that was spent to make all changes became so big.

In the same way, when we have decided to replace a composite primary key for a simple primary key, the work on the app had become bigger too.

In addition, every time that the application has had a change, it has had to be rebuilt and executed, which in terms of time is not worth it.

These are the reasons why last changes we knew that the database need, haven't been reflected on the applications and that's why, even the application is running and we can test most of tables in the database, we can't test every single table, so some tables in the application are different than in the database.

Another handicap we have found is that, even though using java Faces is so easy and quick to create a web application, after building it the first time you have to fix issues with foreign keys, otherwise the application doesn't work.

The conclusion after building this application, and having to make changes in every single file in it, is that a test application should not give us more work when we need to change anything from our software or database, than the proper product which we are testing. This could happen with a production application, but it is not worth spending more time making changes on a test application than on the product that you are developing. So I don't recommend java Faces to test any database.

5. Conclusions

The original project was bigger than this one. It included the design and construction of a database and a java application that could manage a small economy. However, it was too ambitious for a single TFG, then the first decision was split it in two different parts, the database and the application.

The first conclusion I took from working on this project was that when you have to store a lot of information the relationships have to be given a lot of thought. If there is a bad relationship that needs to be changed, it can have a knock-on effect and change other relationships, so data sometimes needs to be repeated to avoid this from happening. This means that in the application design there will have to be methods to throw SQL queries for each repeated attribute, in order to have more information about an entity. However, this information will less needed, while using the application.

Another conclusion I could take was that when you are designing a database you must not think about Object-oriented programming. Throughout this project, I have done just that, and when you are working with too many similar entities is not worth to split tables, so will appear more relationships between them, because there are more tables.

The third conclusion is that if the purpose of the future application is, at most, to manage a company financial asset and have information to make decisions about them, you don't have to store useless information, used to do other type of studies like sectors and markets.

And the last conclusion is about the testing application and software features. First of all, those generic features, which write code by themselves, make more complex designs than you need. This is what has happened when the testing app. A single change on the database has caused several changes on the applications. Make changes on a testing application can't take more time and work than the time you need to implement and fix issues in the main software you want to test.

6. Annex 1

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;
USE `mydb` ;

-- -----------------------------------------------------
-- Table `mydb`.`Subject`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Subject` (
  `NIF` VARCHAR(9) NOT NULL,
  `Name` DATE NOT NULL,
  `Postal_Address_Id` INT NOT NULL,
  `Fiscal_Address_Id` INT NOT NULL,
  `Phone_Number` VARCHAR(12) NOT NULL,
  `Email` VARCHAR(45) NOT NULL,
  `ClientPassword` LONGTEXT NOT NULL,
  `Unsubscrive_date` DATE NULL DEFAULT NULL,
  PRIMARY KEY (`NIF`),
  INDEX `fk_Subject_Address1_idx` (`Postal_Address_Id` ASC),
  INDEX `Subject_FAddressFK_idx` (`Fiscal_Address_Id` ASC),
  CONSTRAINT `Subject_PAddressFK`
    FOREIGN KEY (`Postal_Address_Id`)
    REFERENCES `mydb`.`Address` (`Address_Id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Subject_FAddressFK`
    FOREIGN KEY (`Fiscal_Address_Id`)
    REFERENCES `mydb`.`Address` (`Address_Id`)
    ON DELETE NO ACTION
```

```
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `mydb`.`Person`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Person` (
  `NIF` VARCHAR(9) NOT NULL,
  `1stSurname` VARCHAR(45) NOT NULL,
  `2ndSurname` VARCHAR(45) NULL,
  `BirthDate` DATE NOT NULL,
  `Civil_status` INT NOT NULL,
  `Spouse` VARCHAR(45) NULL,
  `Children` INT NOT NULL,
  PRIMARY KEY (`NIF`),
  INDEX `fk_nif1` (`NIF` ASC),
  CONSTRAINT `NIF_FK`
    FOREIGN KEY (`NIF`)
    REFERENCES `mydb`.`Subject` (`NIF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `mydb`.`Organization`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Organization` (
  `NIF` VARCHAR(9) NOT NULL,
  `Sector` VARCHAR(45) NOT NULL,
  `CrationDate` DATE NOT NULL,
  PRIMARY KEY (`NIF`),
  INDEX `fk_nif` (`NIF` ASC),
```

```
  CONSTRAINT `NIFFK`
    FOREIGN KEY (`NIF`)
    REFERENCES `mydb`.`Subject` (`NIF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Account`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Account` (
  `Numbering` INT NOT NULL,
  `Account_type` VARCHAR(45) NOT NULL,
  `Sing_up_date` DATE NOT NULL,
  `Unsubscrive_date` DATE Null,
  PRIMARY KEY (`Numbering`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`CurrentAccount`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`CurrentAccount` (
  `IBAN` VARCHAR(24) NOT NULL,
  `Balance` BIGINT NOT NULL,
  `Sing_Up_Date` DATE NOT NULL,
  `Currency` VARCHAR(3) NOT NULL,
  `Banking_entity` INT NOT NULL,
  `Account_numFK` INT NOT NULL,
  PRIMARY KEY (`IBAN`),
  INDEX `Account_numFK_idx` (`Account_numFK` ASC),
  INDEX `CurrencyK_idx` (`Currency` ASC),
  INDEX `Bank_entityFK_idx` (`Banking_entity` ASC),
  CONSTRAINT `Banking_entity`
```

```
    FOREIGN KEY (`Account_numFK`)
    REFERENCES `mydb`.`Banks` (`BankID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION),
  CONSTRAINT `Currency3`
    FOREIGN KEY (`Currency`)
    REFERENCES `mydb`.`CurrencyObject` (`Code`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION),
  CONSTRAINT `Account_numFK`
    FOREIGN KEY (`Account_numFK`)
    REFERENCES `mydb`.`Account` (`Numbering`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`ValuesAccount`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`ValuesAccount` (
  `ValuesAccountCode` VARCHAR(59) NOT NULL, -- It's meant to be CCV & ISIN --
  `CCV` VARCHAR(45) NOT NULL,
  `ISIN` VARCHAR(14) NOT NULL,
  `Balance` INT NOT NULL,
  `Account_numFKvalues` INT NOT NULL,
  `Bank_entity` INT NOT NULL,
  PRIMARY KEY (`CCV`, `ISIN`, `Balance`),
  INDEX `Account_numFK_idx` (`Account_numFKvalues` ASC),
  INDEX `Bank_entityFK_idx` (`Bank_entity` ASC),
  CONSTRAINT `Banking_entity3`
    FOREIGN KEY (`Banking_entity`)
    REFERENCES `mydb`.`Banks` (`BankID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION),
```

```
  CONSTRAINT `Account_numFKvalues`
    FOREIGN KEY (`Account_numFKvalues`)
    REFERENCES `mydb`.`Account` (`Numbering`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Financial_assets_Account`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Financial_assets_Account` (
  `Numbering` VARCHAR(45) NOT NULL,
  `ISIN` VARCHAR(14) NOT NULL,
  `Balance` INT NOT NULL,
  `Banking_entity` INT NOT NULL,
  `Account_numFKfinantial` INT NOT NULL,
  PRIMARY KEY (`Numbering`, `ISIN`, `Balance`),
  INDEX `Account_numFK_idx` (`Account_numFKfinantial` ASC),
  INDEX `Bank_entityFK_idx` (`Banking_entity` ASC),
  CONSTRAINT `Banking_entity1`
    FOREIGN KEY (`Banking_entity`)
    REFERENCES `mydb`.`Banks` (`BankID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION),
  CONSTRAINT `Account_numFKfinantial`
    FOREIGN KEY (`Account_numFKfinantial`)
    REFERENCES `mydb`.`Account` (`Numbering`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Participant_Account`
```

```
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Participant_Account` (
  `Depositary_and_Numbering` VARCHAR(45) NOT NULL,
  `Balance` BIGINT NOT NULL,
  `Banking_entity` INT NOT NULL,
  `Account_numFKparticipant` INT NOT NULL,
  PRIMARY KEY (`Depositary_and_Numbering`),
  INDEX `fk_Participant_Account_Account1_idx` (`Account_numFKparticipant`
ASC),
  INDEX `Bank_entityFK_idx` (`Banking_entity` ASC),
  CONSTRAINT `Banking_entity2`
    FOREIGN KEY (`Banking_entity`)
    REFERENCES `mydb`.`Banks` (`BankID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION),
  CONSTRAINT `Account_numFKparticipant`
    FOREIGN KEY (`Account_numFKparticipant`)
    REFERENCES `mydb`.`Account` (`Numbering`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `mydb`.`Product_Type`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Product_Type` (
  `ProductType_ID` INT NOT NULL,
  `Description1` VARCHAR(100) NOT NULL,
  `Description2` VARCHAR(100) NOT NULL,
  `Description3` VARCHAR(100) NOT NULL,
  PRIMARY KEY (`ProductType_ID`))
ENGINE = InnoDB;



-- -----------------------------------------------------
```

```
-- Table `mydb`.`CurrencyObject`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`CurrencyObject` (
  `Code` VARCHAR(3) NOT NULL,
  `Name` VARCHAR(200) NOT NULL,
  `Value` VARCHAR(100) NULL,
  `Issuing_entity` VARCHAR(50) NOT NULL,
  `Sector` INT NOT NULL,
  `Instrument_class` VARCHAR(45) NOT NULL,
  `Product_type` INT NOT NULL,
  `Quoted` TINYINT NOT NULL,
  `Currency` VARCHAR(3) NOT NULL,
  `Issuing_date` DATE NULL,
  `Counterpart` VARCHAR(50) NULL,
  PRIMARY KEY (`Code`),
  INDEX `ProductType_FK_idx` (`Product_type` ASC),
  CONSTRAINT `ProductType_FK`
    FOREIGN KEY (`Product_type`)
    REFERENCES `mydb`.`Product_Type` (`ProductType_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Address`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Address` (
  `Address_Id` INT NOT NULL,
  `Road_type` VARCHAR(45) NOT NULL,
  `Road_Name` VARCHAR(45) NOT NULL,
  `PostalCode` INT NULL,
  `City` VARCHAR(45) NULL,
  `State/Proveince` VARCHAR(45) NULL,
  `Country` VARCHAR(45) NULL,
```

```
  `Number` VARCHAR(45) NULL,
  `Floor` INT NULL,
  `Door` VARCHAR(4) NULL,
  PRIMARY KEY (`Address_Id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`CurrencyTransactions`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`CurrencyTransactions` (
  `Movement_Id` INT NOT NULL AUTO_INCREMENT,
  `IBAN_Account_Related` VARCHAR(24) NOT NULL,
  `IBAN_Account_Related2` VARCHAR(24) NULL,
  `Currency1` VARCHAR(3) NOT NULL,
  `Currency2` VARCHAR(3) NULL,
  `Ammount` DECIMAL NOT NULL,
  `Price` INT NOT NULL,
  `Commissions` FLOAT NOT NULL,
  `Cash_transaction` TINYINT NOT NULL,
  `Transaction_date` DATE NOT NULL,
  `Value_date` DATE NULL,
  `Start_forward_date` DATE NULL,
  `Final_forward_date` DATE NULL,
  `Transaction_author` VARCHAR(9) NOT NULL,
  `Transaction_owner` VARCHAR(45) NOT NULL,
  `Transaction_concept` VARCHAR(45) NULL,
  `Transaction_Origin` INT NULL,
  `Transaction_type` VARCHAR(4) NOT NULL,
  PRIMARY KEY (`Movement_Id`),
  INDEX `IBAN` (`IBAN_Account_Related` ASC),
  INDEX `IBAN2FK_idx` (`IBAN_Account_Related2` ASC),
  INDEX `Currency1FK_idx` (`Currency1` ASC),
  INDEX `Currency2FK_idx` (`Currency2` ASC),
  INDEX `Transaction_authorFK_idx` (`Transaction_author` ASC),
```

```
  INDEX `Transaction_ownerFK_idx` (`Transaction_owner` ASC),
  INDEX `Transaction_type_idx` (`Transaction_type` ASC),
  INDEX `Orice_idx` (`Price` ASC),
  CONSTRAINT `Price`
    FOREIGN KEY (`Price`)
    REFERENCES `mydb`.`Prices` (`Price_Code`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Transaction_type`
    FOREIGN KEY (`Transaction_type`)
    REFERENCES `mydb`.`Transaction_types` (`Name`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `IBANFK`
    FOREIGN KEY (`IBAN_Account_Related`)
    REFERENCES `mydb`.`CurrentAccount` (`IBAN`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `IBAN2FK`
    FOREIGN KEY (`IBAN_Account_Related2`)
    REFERENCES `mydb`.`CurrentAccount` (`IBAN`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Currency1FK`
    FOREIGN KEY (`Currency1`)
    REFERENCES `mydb`.`CurrencyObject` (`Code`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Currency2FK`
    FOREIGN KEY (`Currency2`)
    REFERENCES `mydb`.`CurrencyObject` (`Code`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Transaction_authorFK`
    FOREIGN KEY (`Transaction_author`)
```

```
    REFERENCES `mydb`.`Subject` (`NIF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Transaction_ownerFK`
    FOREIGN KEY (`Transaction_owner`)
    REFERENCES `mydb`.`Subject` (`NIF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Market`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Market` (
  `Market` VARCHAR(6) NOT NULL,
  `Country` VARCHAR(45) NULL,
  `Currency` VARCHAR(3) NOT NULL,
  `Settlement_Currency` INT NOT NULL,
  `Market_Description` VARCHAR(100) NOT NULL,
  `ISO_Code10989MIC` VARCHAR(4) NULL,
  `Intarnational_Code` VARCHAR(20) NOT NULL,
  `Derivatives_Market` TINYINT NOT NULL,
  `Market_Tyoe` VARCHAR(29) NOT NULL,
  `Guarantee_Percentage` INT NOT NULL,
  `Closing_Time` TIME NULL,
  `Fee` VARCHAR(40) NULL,
  `CNMV` VARCHAR(426) NOT NULL,
  `Execution_Cost` VARCHAR(29) NULL,
  `Bloomberg_Code` VARCHAR(17) NULL,
  `BIC_Code` VARCHAR(11) NULL,
  `Regulated` TINYINT NOT NULL,
  `Organized` TINYINT NOT NULL,
  PRIMARY KEY (`Market`),
  INDEX `Currency_FK_idx` (`Currency` ASC),
```

```
  CONSTRAINT `Currency_FK`
    FOREIGN KEY (`Currency`)
    REFERENCES `mydb`.`CurrencyObject` (`Code`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `mydb`.`FundObject`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`FundObject` (
    `AssetCode` VARCHAR(22) NOT NULL, -- It is meant to be code &
Quote_market --
    `Code` VARCHAR(14) NOT NULL,
    `Quote_market` VARCHAR(8) NOT NULL,
    `Name` VARCHAR(200) NOT NULL,
    `Value` VARCHAR(100) NULL,
    `Issuing_entity` VARCHAR(50) NOT NULL,
    `Sector` INT NOT NULL,
    `Instrument_class` VARCHAR(2) NOT NULL,
    `Product_type` INT NOT NULL,
    `Quoted` TINYINT NOT NULL,
    `Currency` VARCHAR(3) NOT NULL,
    `Issuing_date` DATE NULL,
    `Managing_entity` VARCHAR(50) NOT NULL,
    `Composition` VARCHAR(45) NULL,
    PRIMARY KEY (`AssetCode`),
    INDEX `Market_FK_idx` (`Quote_market` ASC),
    INDEX `Currency_FK_idx` (`Currency` ASC),
    INDEX `ProductType_FK_idx` (`Product_type` ASC),
    CONSTRAINT `Market_FKmarket`
    FOREIGN KEY (`Quote_market`)
        REFERENCES `mydb`.`Market` (`Market`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
```

```
    CONSTRAINT `Currency_FKmarket`
    FOREIGN KEY (`Currency`)
        REFERENCES `mydb`.`CurrencyObject` (`Code`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
    CONSTRAINT `ProductType_FKmarket`
    FOREIGN KEY (`Product_type`)
        REFERENCES `mydb`.`Product_Type` (`ProductType_ID`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION
)  ENGINE=INNODB;



-- -----------------------------------------------------
-- Table `mydb`.`FundTransactions`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`FundTransactions` (
  `Transaction_Code` VARCHAR(45) NOT NULL AUTO_INCREMENT,
  `Fund_code` VARCHAR(22) NOT NULL,
  `IBAN_Related` VARCHAR(24) NOT NULL,
  `Depositary_and_NumberingRelated` VARCHAR(45) NOT NULL,
  `Number_of_holdings` INT NOT NULL,
  `Value` FLOAT NOT NULL,
  `Transaction_date` DATE NOT NULL,
  `Value_date` DATE NOT NULL,
  `Settlement_price` FLOAT NOT NULL,
  `Merchant_entity` VARCHAR(50) NOT NULL,
  `Transaction_author` VARCHAR(9) NOT NULL,
  `Transaction_owner` VARCHAR(9) NOT NULL,
  PRIMARY KEY (`Transaction_Code`),
  INDEX `CA_Related_idx` (`IBAN_Related` ASC),
  INDEX `CCV_Related_idx` (`Depositary_and_NumberingRelated` ASC),
  INDEX `Fund_codeFK_idx` (`Fund_code` ASC),
  INDEX `Transaction_author_idx` (`Transaction_author` ASC),
  INDEX `Transaction_owner_idx` (`Transaction_owner` ASC),
```

```
  INDEX `Settelment_price_idx` (`Settelment_price` ASC),
  CONSTRAINT `Settelment_priceFKfundT`
    FOREIGN KEY (`Settelment_price`)
    REFERENCES `mydb`.`Prices` (`Price_Code`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `CA_RelatedFKfundT`
    FOREIGN KEY (`IBAN_Related`)
    REFERENCES `mydb`.`CurrentAccount` (`IBAN`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Depositary_and_NumberingRelatedFK`
    FOREIGN KEY (`Depositary_and_NumberingRelated`)
    REFERENCES `mydb`.`Participant_Account` (`Depositary_and_Numbering`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Fund_codeFKfund`
    FOREIGN KEY (`Fund_code`)
    REFERENCES `mydb`.`FundObject` (`AssetCode`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Transaction_authorFKfund`
    FOREIGN KEY (`Transaction_author`)
    REFERENCES `mydb`.`Subject` (`NIF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Transaction_ownerFKfund`
    FOREIGN KEY (`Transaction_owner`)
    REFERENCES `mydb`.`Subject` (`NIF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
```

```
-- Table `mydb`.`Fixed_incomeObject`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Fixed_incomeObject` (
  `AssetCode` VARCHAR(22) NOT NULL, -- It's meant to be Code & Quote_market -
-
  `Code` VARCHAR(14) NOT NULL,
  `Quote_market` VARCHAR(8) NOT NULL,
  `Name` VARCHAR(200) NOT NULL,
  `Value` VARCHAR(100) NULL,
  `Issuing_entity` VARCHAR(45) NOT NULL,
  `Sector` INT NOT NULL,
  `Instrument_class` VARCHAR(2) NOT NULL,
  `Product_type` INT NOT NULL,
  `Quoted` TINYINT NOT NULL,
  `Currency` VARCHAR(3) NOT NULL,
  `Issuing_date` DATE NULL,
  `Accrual_date` DATE NULL,
  `First_cupon_date` DATE NULL,
  `Expiration_date` DATE NOT NULL,
  `Nominal` FLOAT NOT NULL,
  `Interest_rate` FLOAT NOT NULL,
  `Payday` DATE NOT NULL,
  `Days_of_calculation` INT NOT NULL,
  `Nominal_reduction` FLOAT NULL,
  `Counterpart` VARCHAR(50) NULL,
  `Physic_title` TINYINT NOT NULL,
  `First_call` DATE NULL,
  `PUT` DATE NULL,
  PRIMARY KEY (`AssetCode`),
  INDEX `Market_FK_idx` (`Quote_market` ASC),
  INDEX `ProductType_FK_idx` (`Product_type` ASC),
  INDEX `Currency_idx` (`Currency` ASC),
  CONSTRAINT `Market_FKfixed`
    FOREIGN KEY (`Quote_market`)
    REFERENCES `mydb`.`Market` (`Market`)
    ON DELETE NO ACTION
```

```
    ON UPDATE NO ACTION,
  CONSTRAINT `ProductType_FKfixed`
    FOREIGN KEY (`Product_type`)
    REFERENCES `mydb`.`Product_Type` (`ProductType_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Currencyfixed`
    FOREIGN KEY (`Currency`)
    REFERENCES `mydb`.`CurrencyObject` (`Code`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Fixed_IncomeTransactions`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Fixed_IncomeTransactions` (
  `Transaction_code` INT NOT NULL,
  `ISINrelated` VARCHAR(22) NOT NULL,
  `IBAN_Related` VARCHAR(34) NOT NULL,
  `CCV_Related` VARCHAR(59) NOT NULL,
  `Ammount` INT NOT NULL,
  `Transaction_date` DATE NOT NULL,
  `Date_value` DATE NOT NULL,
  `Nominal` INT NOT NULL,
  `Broker` VARCHAR(50) NULL,
  `Ex_cupon_price` FLOAT NULL,
  `Run_cupon` FLOAT NULL,
  `Total_cash` FLOAT NOT NULL,
  `Comissions` FLOAT NOT NULL,
  `Total_paid` DECIMAL NOT NULL,
  `Transaction_author` VARCHAR(9) NOT NULL,
  `Transaction_owner` VARCHAR(9) NOT NULL,
  PRIMARY KEY (`Transaction_code`),
```

```
  INDEX `CA_Related_idx` (`IBAN_Related` ASC),
  INDEX `CCV_RelatedFK_idx` (`CCV_Related` ASC),
  INDEX `ISISNrelatedFK_idx` (`ISINrelated` ASC),
  INDEX `Transaction_authorFK_idx` (`Transaction_author` ASC),
  INDEX `Transaction_ownerFK_idx` (`Transaction_owner` ASC),
  INDEX `NominalFK_idx` (`Nominal` ASC),
  CONSTRAINT `NominalFKfixedT`
    FOREIGN KEY (`Nominal`)
    REFERENCES `mydb`.`Prices` (`Price_Code`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `CA_RelatedFKfixedT`
    FOREIGN KEY (`IBAN_Related`)
    REFERENCES `mydb`.`CurrentAccount` (`IBAN`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `CCV_RelatedFKfixedT`
    FOREIGN KEY (`CCV_Related`)
    REFERENCES `mydb`.`ValuesAccount` (`ValuesAccountCode`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `ISINrelatedFK`
    FOREIGN KEY (`ISINrelated`)
    REFERENCES `mydb`.`Fixed_incomeObject` (`AssetCode`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Transaction_authorFKfixedT`
    FOREIGN KEY (`Transaction_author`)
    REFERENCES `mydb`.`Subject` (`NIF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Transaction_ownerFKfixedT`
    FOREIGN KEY (`Transaction_owner`)
    REFERENCES `mydb`.`Subject` (`NIF`)
    ON DELETE NO ACTION
```

```
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`DerivativeObject`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`DerivativeObject` (
  `AssetCode` VARCHAR(22) NOT NULL,
  `Code` VARCHAR(14) NOT NULL,
  `Quote_market` VARCHAR(8) NOT NULL,
  `Name` VARCHAR(200) NOT NULL,
  `Value` VARCHAR(100) NULL,
  `Issuing_entity` VARCHAR(50) NOT NULL,
  `Sector` INT NOT NULL,
  `Instrument_class` VARCHAR(2) NOT NULL,
  `Product_type` INT NOT NULL,
  `Quoted` TINYINT NOT NULL,
  `Currency` VARCHAR(3) NOT NULL,
  `Issuing_date` DATE NOT NULL,
  `Expiration_date` DATE NULL,
  `Nominal` FLOAT NOT NULL,
  PRIMARY KEY (`Code`, `Quote_market`),
  INDEX `Market_FK_idx` (`Quote_market` ASC),
  INDEX `ProductType_FK_idx` (`Product_type` ASC),
  INDEX `Currency_FK_idx` (`Currency` ASC),
  CONSTRAINT `Market_FKdeivativeO`
    FOREIGN KEY (`Quote_market`)
    REFERENCES `mydb`.`Market` (`Market`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `ProductType_FKdeivativeO`
    FOREIGN KEY (`Product_type`)
    REFERENCES `mydb`.`Product_Type` (`ProductType_ID`)
    ON DELETE NO ACTION
```

```
    ON UPDATE NO ACTION,
  CONSTRAINT `Currency_FKdeivativeO`
    FOREIGN KEY (`Currency`)
    REFERENCES `mydb`.`CurrencyObject` (`Code`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`DerivativeTransactions`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`DerivativeTransactions` (
  `Transaction_Code` INT NOT NULL AUTO_INCREMENT,
  `Product_code` VARCHAR(22) NULL,
  `IBAN_Related` INT NOT NULL,
  `FA_Account_Related` INT NOT NULL,
  `Price` INT NOT NULL,
  `Titles_ammount` INT NOT NULL,
  `Maturity_of_the_asset` DATE NOT NULL,
  `Transaction_date` DATE NOT NULL,
  `Value_date` DATE NOT NULL,
  `Broker` VARCHAR(50) NULL,
  `Underlying_ammount` FLOAT NULL,
  `Total_cash` FLOAT NOT NULL,
  `Comissions` FLOAT NOT NULL,
  `Total_paid` FLOAT NOT NULL,
  `Transaction_author` VARCHAR(9) NOT NULL,
  `Transaction_owner` VARCHAR(9) NOT NULL,
  PRIMARY KEY (`CodTransaction_c`),
  INDEX `IBAN` (`IBAN_Related` ASC),
  INDEX `FA_Account_RelatedFK_idx` (`FA_Account_Related` ASC),
  INDEX `DerivativeCodeFK_idx` (`Derivative_codeFKderivativeT` ASC),
  INDEX `Transaction_authorFK_idx` (`Transaction_author` ASC),
  INDEX `Transaction_owner_idx` (`Transaction_owner` ASC),
```

```
  INDEX `Transaction_price_idx` (`Price` ASC),
  CONSTRAINT `Price`
    FOREIGN KEY (`Price`)
    REFERENCES `mydb`.`Prices` (`Price_Code`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `IBAN_Account_Related`
    FOREIGN KEY (`IBANt_RelatedFKderivativeT`)
    REFERENCES `mydb`.`CurrentAccount` (`IBAN`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `FA_Account_Related`
    FOREIGN KEY (`FA_Account_Related`)
    REFERENCES `mydb`.`Financial_assets_Account` (`Numbering`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `ISIN`
    FOREIGN KEY (`Product_code`)
    REFERENCES `mydb`.`DerivativeObject` (`AssetCode`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Transaction_authorFKderivativeT`
    FOREIGN KEY (`Transaction_author`)
    REFERENCES `mydb`.`Subject` (`NIF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Transaction_ownerFKderivativeT`
    FOREIGN KEY (`Transaction_owner`)
    REFERENCES `mydb`.`Subject` (`NIF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
```

```
-- Table `mydb`.`ReferenceRegister`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`ReferenceRegister` (
  `ReferenceRegister_Id` VARCHAR(45) NOT NULL,
  `Account_numFKreference` INT NOT NULL,
  PRIMARY KEY (`ReferenceRegister_Id`),
  INDEX `Account_numFKreference_idx` (`Account_numFKreference` ASC),
  CONSTRAINT `Account_numFKreference`
    FOREIGN KEY (`Account_numFK`)
    REFERENCES `mydb`.`Account` (`Numbering`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `mydb`.`PropertyObject`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`PropertyObject` (
  `Code` VARCHAR(20) NOT NULL,
  `Name` VARCHAR(200) NOT NULL,
  `Value` VARCHAR(100) NULL,
  `Issuing_entity` VARCHAR(50) NULL,
  `Sector` INT NOT NULL,
  `Instrument_class` VARCHAR(2) NOT NULL,
  `Product_type` INT NOT NULL,
  `Currency` VARCHAR(3) NOT NULL,
  `Issuing_date` DATE NULL,
  `LandRegistry_value` FLOAT NULL,
  PRIMARY KEY (`Code`),
  INDEX `ProductType_FK_idx` (`Product_type` ASC),
  INDEX `Currency_FK_idx` (`Currency` ASC),
  CONSTRAINT `ProductType_FKpropertyO`
    FOREIGN KEY (`Product_type`)
    REFERENCES `mydb`.`Product_Type` (`ProductType_ID`)
```

```
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Currency_FKpropertyO`
    FOREIGN KEY (`Currency`)
    REFERENCES `mydb`.`CurrencyObject` (`Code`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`PropertyTransactions`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`PropertyTransactions` (
  `Transaction_code` INT NOT NULL AUTO_INCREMENT,
  `Land_Registry_code` VARCHAR(20) NOT NULL,
  `IBAN_Related` VARCHAR(24) NOT NULL,
  `ReferenceRegister_Id` VARCHAR(45) NOT NULL,
  `Transaction_date` DATE NOT NULL,
  `Price` INT NOT NULL,
  `Transaction_author` VARCHAR(9) NOT NULL,
  `Transaction_owner` VARCHAR(9) NOT NULL,
  PRIMARY KEY (`Transaction_code`),
  INDEX `IBAN_Related_idx` (`IBAN_Related` ASC),
  INDEX `fk_Propoerty_ReferenceRegister1_idx` (`ReferenceRegister_Id` ASC),
  INDEX `Land_Registry_codeFK_idx` (`Land_Registry_code` ASC),
  INDEX `Transaction_authorFK_idx` (`Transaction_author` ASC),
  INDEX `Transaction_ownerFK_idx` (`Transaction_owner` ASC),
  INDEX `Transaction_price_idx1` (`Price` ASC),
  CONSTRAINT `Price`
    FOREIGN KEY (`Price`)
    REFERENCES `mydb`.`Prices` (`Price_Code`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `IBAN_RelatedFKpropertyT`
```

```
    FOREIGN KEY (`IBAN_Related`)
    REFERENCES `mydb`.`CurrentAccount` (`IBAN`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `RefReg_IdFKpropertyT`
    FOREIGN KEY (`ReferenceRegister_Id`)
    REFERENCES `mydb`.`ReferenceRegister` (`ReferenceRegister_Id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Land_Registry_codeFKpropertyT`
    FOREIGN KEY (`Land_Registry_code`)
    REFERENCES `mydb`.`PropertyObject` (`Code`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Transaction_authorFKpropertyT`
    FOREIGN KEY (`Transaction_author`)
    REFERENCES `mydb`.`Subject` (`NIF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Transaction_ownerFKpropertyT`
    FOREIGN KEY (`Transaction_owner`)
    REFERENCES `mydb`.`Subject` (`NIF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `mydb`.`Variable_incomeObject`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Variable_incomeObject` (
  `AssetCode` VARCHAR(22) NOT NULL, -- It's meant to be Code & Quote_market -
-
  `Code` VARCHAR(14) NOT NULL,
  `Quote_market` VARCHAR(8) NOT NULL,
  `Name` VARCHAR(200) NOT NULL,
```

```
  `Value` VARCHAR(100) NULL,
  `Issuing_entity` VARCHAR(50) NOT NULL,
  `Sector` INT NOT NULL,
  `Instrument_class` VARCHAR(2) NOT NULL,
  `Product_type` INT NOT NULL,
  `Quoted` TINYINT NOT NULL,
  `Currency` VARCHAR(3) NOT NULL,
  `Issuing_date` DATE NULL,
  `Nominal` FLOAT NOT NULL,
  `Physic_title` TINYINT NOT NULL,
  PRIMARY KEY (`Code`, `Quote_market`),
  INDEX `Market_FK_idx` (`Quote_market` ASC),
  INDEX `Currency_FK_idx` (`Currency` ASC),
  INDEX `ProductType_FK_idx` (`Product_type` ASC),
  CONSTRAINT `Market_FKvariableO`
    FOREIGN KEY (`Quote_market`)
    REFERENCES `mydb`.`Market` (`Market`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Currency_FKvariableO`
    FOREIGN KEY (`Currency`)
    REFERENCES `mydb`.`CurrencyObject` (`Code`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `ProductType_FKvariableO`
    FOREIGN KEY (`Product_type`)
    REFERENCES `mydb`.`Product_Type` (`ProductType_ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Variable_IncomeTransactions`
-- -----------------------------------------------------
```

```
CREATE TABLE IF NOT EXISTS `mydb`.`Variable_IncomeTransactions` (
  `Transaction_code` INT NOT NULL AUTO_INCREMENT,
  `ISINrelated` VARCHAR(22) NOT NULL,
  `IBAN_Related` VARCHAR(25) NOT NULL,
  `CCV_Related` VARCHAR(59) NOT NULL,
  `Transaction_date` DATE NOT NULL,
  `Date_value` DATE NOT NULL,
  `Number_of_shares` INT NOT NULL,
  `Broker` VARCHAR(50) NULL,
  `Price` INT NOT NULL,
  `Comissions` FLOAT NOT NULL,
  `Total_paid` FLOAT NOT NULL,
  `Transaction_author` VARCHAR(9) NOT NULL,
  `Transaction_owner` VARCHAR(9) NOT NULL,
  PRIMARY KEY (`Transaction_code`),
  INDEX `IBAN_Related_idx` (`IBAN_Related` ASC),
  INDEX `CCV_Related_idx` (`CCV_Related` ASC),
  INDEX `Product_code_idx` (`Product_code` ASC),
  INDEX `Transaction_authorFK_idx` (`Transaction_author` ASC),
  INDEX `Transaction_ownerFK_idx` (`Transaction_owner` ASC),
  INDEX `Transaction_price_idx2` (`Price` ASC),
  CONSTRAINT `Price`
    FOREIGN KEY (`Price`)
    REFERENCES `mydb`.`Prices` (`Price_Code`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `IBAN_RelatedFKvariableT`
    FOREIGN KEY (`IBAN_Related`)
    REFERENCES `mydb`.`CurrentAccount` (`IBAN`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `CCV_RelatedFKvariableT`
    FOREIGN KEY (`CCV_Related`)
    REFERENCES `mydb`.`ValuesAccount` (`ValuesAccountCode`)
    ON DELETE NO ACTION
```

```
    ON UPDATE NO ACTION,
  CONSTRAINT `Product_codeFKvariableT`
    FOREIGN KEY (`ISINrelated`)
    REFERENCES `mydb`.`Variable_incomeObject` (`AssetCode`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Transaction_authorFKvariableT`
    FOREIGN KEY (`Transaction_author`)
    REFERENCES `mydb`.`Subject` (`NIF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Transaction_ownerFKvariableT`
    FOREIGN KEY (`Transaction_owner`)
    REFERENCES `mydb`.`Subject` (`NIF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Participation`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Participation` (
  `Subject_NIF` VARCHAR(9) NOT NULL,
  `Account_Numbering` INT NOT NULL,
  `Participation` DECIMAL NOT NULL,
  PRIMARY KEY (`Subject_NIF`, `Account_Numbering`),
  INDEX `fk_Subject_has_Account_Account1_idx` (`Account_Numbering` ASC),
  INDEX `fk_Subject_has_Account_Subject1_idx` (`Subject_NIF` ASC),
  CONSTRAINT `fk_Subject_has_Account_Subject1`
    FOREIGN KEY (`Subject_NIF`)
    REFERENCES `mydb`.`Subject` (`NIF`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Subject_has_Account_Account1`
```

```
    FOREIGN KEY (`Account_Numbering`)
    REFERENCES `mydb`.`Account` (`Numbering`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Banks`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Banks` (
  `BankID` INT NOT NULL,
  `Bank_Name` VARCHAR(100) NOT NULL,
  `Address` INT NULL,
  PRIMARY KEY (`BankID`),
  INDEX `Address_FK_idx` (`Address` ASC),
  CONSTRAINT `Address_FK`
    FOREIGN KEY (`Address`)
    REFERENCES `mydb`.`Address` (`Address_Id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Prices`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Prices` (
  `Price_Code` INT NOT NULL AUTO_INCREMENT,
  `AssetCode` VARCHAR(14) NOT NULL,
  `Quote_market` VARCHAR(45) NOT NULL,
  `Date` DATE NOT NULL,
  `Currency`VARCHAR(3) NOT NULL,
  `Price` DOUBLE NOT NULL,
  PRIMARY KEY (`Price_Code`),
```

```
  INDEX `fk_Prices_6_idx`(`Quote_market` ASC),
  INDEX `Currency_idx`(`Currency` ASC),
  CONSTRAINT `Currency4`
    FOREIGN KEY (`Currency`)
    REFERENCES `mydb`.`CurrencyObject` (`Code`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Prices_6`
    FOREIGN KEY (`Quote_market`)
    REFERENCES `mydb`.`Market` (`Market`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Transaction_types`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Transaction_types` (
  `Name` VARCHAR(4) NOT NULL,
  `Family_affected` TINYINT NULL,
  `Family_sign` TINYINT NULL,
  `Professional_affected` TINYINT NULL,
  `Other_Accounts_affected` TINYINT NULL,
  `Tax_Repercusion` TINYINT NULL,
  `Description` VARCHAR(200) NULL,
  PRIMARY KEY (`Name`))
ENGINE = InnoDB;

USE `mydb`;

DELIMITER $$
USE `mydb`$$
```

```
CREATE DEFINER = CURRENT_USER TRIGGER
`mydb`.`CurrencyTransactions_AFTER_INSERT` AFTER INSERT ON
`CurrencyTransactions` FOR EACH ROW

BEGIN


update CurrentAccount set CurrentAccount.Balance = CurrentAccount.Balance +
new.Ammount where CurrentAccount.IBAN = new.IBAN_Account_Related ;

END$$


USE `mydb`$$

CREATE DEFINER = CURRENT_USER TRIGGER
`mydb`.`CurrencyTransactions_AFTER_INSERT` AFTER DELETE ON
`CurrencyTransactions` FOR EACH ROW

BEGIN


update CurrentAccount set CurrentAccount.Balance = CurrentAccount.Balance -
new.Ammount where CurrentAccount.IBAN = new.IBAN_Account_Related ;

END$$

DELIMITER ;


SET SQL_MODE=@OLD_SQL_MODE;

SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;

SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

7. References

[1] "Efinance management," [Online]. Available: https://efinancemanagement.com/sources-of-finance/financial-instruments. [Accessed 13 June 2019].

[2] USLegal TM, "uslegal.com," [Online]. Available: https://definitions.uslegal.com/c/cash-instruments/. [Accessed 16 June 2019].

[3] "Cambridge Dictionary," [Online]. Available: https://dictionary.cambridge.org/dictionary/english/currency. [Accessed 13 June 2019].

[4] Investopedia, "invertopedia.com," [Online]. Available: https://www.investopedia.com/terms/i/investment-fund.asp. [Accessed 16 June 2019].

[5] Investopedia, "investopedia.com," [Online]. Available: https://www.investopedia.com/terms/f/fixed-incomesecurity.asp. [Accessed 17 June 2019].

[6] Investopedia, "investopedia.com," [Online]. Available: https://www.investopedia.com/terms/i/investment-property.asp. [Accessed 17 June 2019].

[7] M.-z. TM, "money-zine.com," [Online]. Available: https://www.money-zine.com/definitions/investing-dictionary/variable-income-securities/. [Accessed 16 June 2019].

[8] USLegal, "uslegal.com," [Online]. Available: https://definitions.uslegal.com/d/derivative-instruments/. [Accessed 17 June 2019].

[9] Investopedia, "investopedia.com," [Online]. Available: https://www.investopedia.com/terms/a/accruals.asp. [Accessed 22 June 2019].

[10] Investopedia, “investopedia.com,” [Online]. Available: https://www.investopedia.com/terms/c/counterparty.asp. [Accessed 22 June 2019].

[11] Investopedia, “investopedia.com,” [Online]. Available: https://www.investopedia.com/terms/m/maturitydate.asp. [Accessed 22 June 2019].

[12] Investopedia, “investopedia,” [Online]. Available: https://www.investopedia.com/terms/e/expirationdate.asp. [Accessed 22 June 2019].

[13] F. dictionary, “https://financial-dictionary.thefreedictionary.com,” [Online]. Available: https://financial-dictionary.thefreedictionary.com/first+call+date. [Accessed 22 June 2019].

[14] Investopedia, “investopedia.com,” [Online]. Available: https://www.investopedia.com/terms/a/assetclasses.asp. [Accessed 22 June 2019].

[15] NASDAQ, “nasdaq.com,” [Online]. Available: https://www.nasdaq.com/investing/glossary/d/date-of-issue. [Accessed 22 June 2019].

[16] Bizfluent, “bizfluent.com,” [Online]. Available: https://bizfluent.com/info-8635730-issuing-entity.html. [Accessed 18 June 2019].

[17] Investopedia, “investopedia.com,” [Online]. Available: https://www.investopedia.com/terms/i/interestrate.asp. [Accessed 22 June 2019].

[18] Investopedia, “investopedia.com,” [Online]. Available: https://www.investopedia.com/terms/f/funds-management.asp. [Accessed 22 June 2019].

[19] Investopedia, “investopedia.com,” [Online]. Available: https://www.investopedia.com/terms/p/principal.asp. [Accessed 22 June 2019].

[20] Investopedia, “investopedia.com,” [Online]. Available: https://www.investopedia.com/terms/p/principal-reduction.asp. [Accessed 22 June 2019].

[21] Investopedia, “investopedia.com,” [Online]. Available: https://www.investopedia.com/terms/p/put.asp. [Accessed 22 June 2019].

[22] Investopedia, “investopedia.com,” [Online]. Available: https://www.investopedia.com/terms/f/financial-market.asp. [Accessed 22 June 2019].

[23] B. dictionary, “businessdictionary.com,” [Online]. Available: http://www.businessdictionary.com/definition/economic-sector.html. [Accessed 22 June 2019].

[24] E. online, “https://www.economicsonline.co.uk/,” [Online]. Available: https://www.economicsonline.co.uk/Definitions/Current_account.html. [Accessed 24 June 2019].

[25] Investopedia, “investopedia.com,” [Online]. Available: https://www.investopedia.com/terms/i/iban.asp. [Accessed 24 June 2019].

[26] Transferwise, “transferwise.com,” [Online]. Available: https://transferwise.com/gb/iban/uk/monzo-bank. [Accessed 24 June 2019].

[27] Investopedia, “investopedia.com,” [Online]. Available: https://www.investopedia.com/terms/a/accountbalance.asp. [Accessed 24 June 2019].

[28] Anna-web, “anna-web.org/,” [Online]. Available: https://www.anna-web.org/standards/isin-iso-6166/. [Accessed 24 June 2019].

[29] G. d. España, “http://www.catastro.meh.es/,” [Online]. Available: http://www.catastro.meh.es/esp/referencia_catastral_1.asp. [Accessed 24 June 2019].

[30] B. dictionary, “businessdictionary.com/,” [Online]. Available: http://www.businessdictionary.com/definition/transaction.html. [Accessed 24 June 2019].

[31] Investopedia, “investopedia.com,” [Online]. Available: https://www.investopedia.com/terms/p/payee.asp. [Accessed 25 June 2019].

[32] Investopedia, “investopedia.com,” [Online]. Available: https://www.investopedia.com/terms/f/forward-start-price.asp. [Accessed 30 June 2019].

[33] Investopedia, “investopedia.com,” [Online]. Available: https://www.investopedia.com/terms/s/settlementprice.asp. [Accessed 26 June 2019].

[34] Investopedia, “investopedia.com,” [Online]. Available: https://www.investopedia.com/terms/b/broker.asp. [Accessed 25 June 2019].

[35] Investopedia, “investopedia.com,” [Online]. Available: https://www.investopedia.com/terms/u/underlying.asp. [Accessed 27 June 2019].

[36] I. Brokers, “https://ibkr.info,” [Online]. Available: https://ibkr.info/glossary/1283. [Accessed 30 June 2019].

[37] Investopedia, “investopedia.com,” [Online]. Available: https://www.investopedia.com/terms/o/otc.asp. [Accessed 30 June 2019].

[38] Investopedia, “investopedia.com,” [Online]. Available: https://www.investopedia.com/terms/f/financial-guarantee.asp. [Accessed 30 June 2019].

[39] NASDAQ, “nasdaq.com,” [Online]. Available: https://www.nasdaq.com/investing/glossary/e/execution-costs. [Accessed 30 June 2019].

[40] B. LP, “https://www.bloomberg.com/,” [Online]. Available: company/?utm_source=bloomberg-menu&utm_medium=bcom. [Accessed 1 July 2019].

[41] Caixabank, “caixabank.cat,” [Online]. Available: https://www.caixabank.cat/particular/atencioclient/faqs/asistentefaq6_ca.html. [Accessed 1 July 2019].

[42] O. Reference, “https://www.oxfordreference.com/,” [Online]. Available: https://www.oxfordreference.com/view/10.1093/oi/authority.20110803100253883. [Accessed 1 July 2019].

[43] w3schools, “https://www.w3schools.com,” [Online]. Available: https://www.w3schools.com/sql/sql_primarykey.asp. [Accessed 2 July 2019].

[44] w3schools, “https://www.w3schools.com/,” [Online]. Available: https://www.w3schools.com/sql/sql_foreignkey.asp. [Accessed 2 July 2019].

[45] J. point, “https://www.javatpoint.com/,” [Online]. Available: https://www.javatpoint.com/sql-composite-key. [Accessed 2 July 2019].

[46] MySQL.com, “https://www.mysql.com/,” [Online]. Available: https://www.mysql.com/products/workbench/. [Accessed 10 July 2019].

[47] Oracle, “https://www.oracle.com/,” [Online]. Available: https://www.oracle.com/technetwork/middleware/glassfish/overview/index.html. [Accessed 10 July 2019].

[48] MySQLtutorial.org, “http://www.mysqltutorial.org/,” [Online]. Available: http://www.mysqltutorial.org/sql-triggers.aspx. [Accessed 9 July 2019].

[49] J. world, “https://www.javaworld.com/,” [Online]. Available: https://www.javaworld.com/article/3322533/what-is-jsf-introducing-javaserver-faces.html. [Accessed 9 July 2019].